2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

IROS18

10th International workshop on

Planning, Perception and Navigation for Intelligent Vehicles

Full Day Workshop

October 1st, 2018, Madrid, Spain

https://project.inria.fr/ppniv18/

Organizers

Pr Christian Laugier (INRIA, France),
Pr Philippe Martinet (IRCCYN, France),
Pr Urbano Nunes (ISR, Portugal)
Pr M.A. Sotelo (University of Alcalà, Spain)
Pr Christoph Stiller (KIT, Germany)

Contact

Director of Research Philippe Martinet
Inria - CHORALE team
2004 route des Lucioles, 06902 Sophia-Antipolis, FRANCE
Phone: +33 240 376 975, Sec : +33 240 376 934, Fax : +33 240 376 6930
Email: Philippe.Martinet@inria.fr
Home page: http://www-sop.inria.fr/members/Philippe.Martinet/

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Foreword

The purpose of this workshop is to discuss topics related to the challenging problems of autonomous navigation and of driving assistance in open and dynamic environments. Technologies related to application fields such as unmanned outdoor vehicles or intelligent road vehicles will be considered from both the theoretical and technological point of views. Several research questions located on the cutting edge of the state of the art will be addressed. Among the many application areas that robotics is addressing, transportation of people and goods seem to be a domain that will dramatically benefit from intelligent automation. Fully automatic driving is emerging as the approach to dramatically improve efficiency while at the same time leading to the goal of zero fatalities. This workshop will address robotics technologies, which are at the very core of this major shift in the automobile paradigm. Technologies related to this area, such as autonomous outdoor vehicles, achievements, challenges and open questions would be presented. Main topics include: Road scene understanding, Lane detection and lane keeping, Pedestrian and vehicle detection, Detection, tracking and classification, Feature extraction and feature selection, Cooperative techniques, Collision prediction and avoidance, Advanced driver assistance systems, Environment perception, vehicle localization and autonomous navigation, Real-time perception and sensor fusion, SLAM in dynamic environments, Mapping and maps for navigation, Real-time motion planning in dynamic environments, Human-Robot Interaction, Behavior modeling and learning, Robust sensor-based 3D reconstruction, Modeling and Control of mobile robot.

Previously, several workshops were organized in the near same field. The 1st edition PPNIV'07 of this workshop was held in Roma during ICRA'07 (around 60 attendees), the second PPNIV'08 was in Nice during IROS'08 (more than 90 registered people), the third PPNIV'09 was in Saint-Louis (around 70 attendees) during IROS'09, the fourth edition PPNIV'12 was in Vilamoura (over 95 attendees) during IROS'12, the fifth edition PPNIV'13 was in Vilamoura (over 135 attendees) during IROS'13, the sixth edition PPNIV'14 was in Chicago (over 100 attendees) during IROS14, the seventh edition PPNIV'15 was in Hamburg (over 150 attendees) during IROS15, the heigth edition PPNIV'16 was in Rio de Janeiro (over 100 attendees) during ITSC16; and the nineth edition PPNIV17 was in Vancouver during IROS17 (over 170 attendees). This 10th edition has gathered over 350 attendees.

In parallel, we have also organized SNODE'07 in San Diego during IROS'07 (around 80 attendees), MEPPC08 in Nice during IROS'08 (more than 60 registered people), SNODE'09 in Kobe during ICRA'09 (around 70 attendees), RITS'10 in Anchrorage during ICRA'10 (around 35 attendees), PNAVHE11 in San Francisco during the last IROS11 (around 50 attendees), and the last one WMEPC14 in Hong Kong during the last ICRA14 (around 65 attendees),

This workshop is composed with 3 invited talks and 20 selected papers (8 selected for oral presentation and 12 selected for interactive session. One round table has gathered specialist in Autonomous Shuttle and Taxis. Five sessions have been organized:

- Session I: Deep Learning
- Session II: Navigation, Decision, Safety
- Session III: Perception
- Session IV: Interactive session

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

- Session V: Control & Planning

Intended Audience concerns researchers and PhD students interested in mobile robotics, motion and action planning, robust perception, sensor fusion, SLAM, autonomous vehicles, human-robot interaction, and intelligent transportation systems. Some peoples from the mobile robot industry and car industry are also welcome.

This workshop is made in relation with IEEE RAS: RAS Technical Committee on "Autonomous Ground Vehicles and Intelligent Transportation Systems" (http://tab.ieee-ras.org/).

Christian Laugier, Philippe Martinet, Urbano Nunes, Miguel Angel Sotelo and Christoph Stiller

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session I

Deep Learning

- **Title: ISA2: Intelligent Speed Adaptation from appearance**
  Authors: C. Herranz-Perdiguero and R. J. Lopez-Sastre

- **Title: Classification of Point Cloud for Road Scene Understanding with Multiscale Voxel Deep Network**
  Authors: X. Roynard and J.E. Deschaud and F. Goulette

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

ISA^2: Intelligent Speed Adaptation from Appearance

Carlos Herranz-Perdiguero[1] and Roberto J. López-Sastre[1]

***Abstract*— In this work we introduce a new problem named Intelligent Speed Adaptation from Appearance (ISA^2). Technically, the goal of an ISA^2 model is to predict for a given image of a driving scenario the *proper* speed of the vehicle. Note this problem is different from predicting the actual speed of the vehicle. It defines a novel regression problem where the appearance information has to be directly mapped to get a prediction for the speed at which the vehicle should go, taking into account the traffic situation. First, we release a novel dataset for the new problem, where multiple driving video sequences, with the annotated adequate speed per frame, are provided. We then introduce two deep learning based ISA^2 models, which are trained to perform the final regression of the proper speed given a test image. We end with a thorough experimental validation where the results show the level of difficulty of the proposed task. The dataset and the proposed models will all be made publicly available to encourage much needed further research on this problem.**

I. INTRODUCTION

For years, speed has been recognized as one of the three main contributing factors to deaths on our roads. In fact, 72 % of road traffic accidents in the city could be prevented with an adequate vehicle speed, according to the MAPFRE Foundation [1]. Furthermore, the European Transport Safety Council (ETSC) claims that speed is the cause of the death of 500 people every week on European roads [2]. So, to control the speed of our vehicles, using an Intelligent Speed Adaptation (ISA) system, should be a high-priority research line.

A research by the Norwegian Institute for Transport Economics [3] advocates the benefits of an ISA system, which the study found to be the most effective solution in saving lives. Some studies of the ETSC reveal that the adoption of the ISA technology is expected to reduce collisions by 30% and deaths by 20% [4].

Off-the-shelf ISA solutions use a speed traffic sign recognition module, and/or GPS-linked speed limit data to inform the drivers of the current speed limit of the road or highway. However, these solutions have the following limitations. First, GPS information is inaccurate and may not be correctly updated. For example, an ISA model based only on GPS information would have difficulties in certain urban scenes with poor satellite visibility, or in distinguishing whether the vehicle is in a highway lane or on the nearby service road, where the speed limit has to be drastically reduced. It is true that a speed traffic sign recognition module can mitigate some of these problems, but for doing so we need to guarantee the visibility of the signs. Second, they provide only the speed limit of the road, but not the speed *appropriate* to the actual traffic situation.

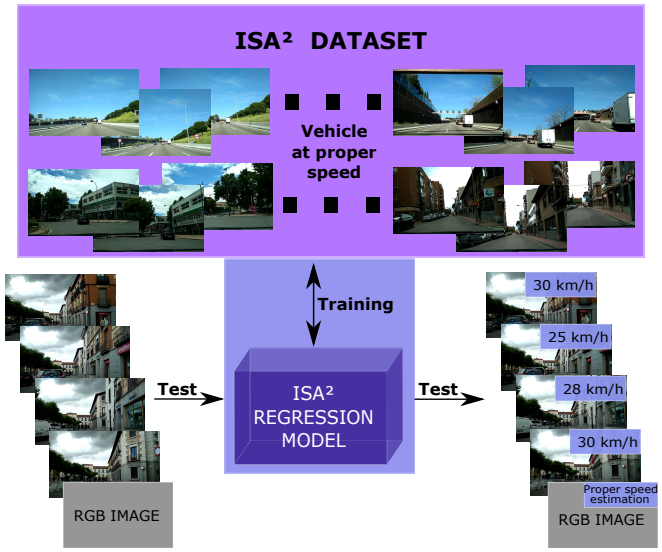


Fig. 1: ISA^2 problem. An ISA^2 model must be able to perform a regression of the *adequate* speed of the vehicle, inferring it just using the appearance information of the image. It has to be trained on video sequences providing the proper speed for the traffic situation, to be able to provide an estimation of the adequate speed on test images.

To address all these limitations, in this paper we propose a new paradigm for the ISA models, called **ISA** from **A**ppearance, or ISA^2. Technically, as it is shown in Figure 1, we introduce the idea of learning a regression function able to map the images to a speed adequate to the traffic situation. For doing so, we need to train and evaluate the ISA^2 solutions using a dataset with video sequences that show a driving behaviour that is appropriate to the real traffic situation. The proposed problem is actually very challenging. Could a human, from a single image, discern between whether a vehicle should go at 80 or 110 km/h on a motorway according to the actual traffic?

The main contributions of our work are as follows:

1) To the best of our knowledge, we propose for the first time the novel problem of inferring the *adequate* speed of a vehicle from just an image.

*This work is supported by project PREPEATE, with reference TEC2016-80326-R, of the Spanish Ministry of Economy, Industry and Competitiveness. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a GPU used for this research. Cloud computing resources were kindly provided through a Microsoft Azure for Research Award.

[1]The authors are with GRAM research group, Department of Signal Theory and Communications, University of Alcalá, 28805, Alcalá de Henares, Spain c.herranz,@edu.uah.es, robertoj.lopez@.uah.es

2) We introduce two deep learning based ISA^2 models, which are trained to perform the final regression of the proper speed for the vehicle. One consists in learning a deep network to directly perform the speed regression. The other approach is based on a deep learning model to obtain a semantic segmentation of the traffic scene. We then combine this output with a spatial pyramid pooling strategy to build the features used to learn the regressor for the proper speed.
3) We also release a novel dataset for the new ISA^2 problem, where the proposed models are evaluated. We conduct an extensive set of experiments and show that our ISA^2 solutions can report an error for the prediction of the speed lower than 6 km/h.

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III we describe the ISA^2 dataset and the evaluation protocol. Our ISA^2 models are detailed in Section IV. We evaluate our models, and analyze their performance in Section V. We conclude in Section VI.

II. RELATED WORK

Although being able to estimate the appropriate speed for a vehicle is a key task for the automotive industry, which year after year is increasing the budget for R&D projects in its pursuit to achieve a fully autonomous vehicle, there are no previous works that seek to predict this speed just using images or visual information.

In the literature, we can find some works that deal with the different problem of learning a generic driving model, *e.g.* [5], [6], [7].

Probably, the closest works we can find to the problem we are trying to solve, focus on estimating the *actual* speed of a vehicle, which is a different problem anyhow. Several techniques have been proposed for this purpose, from the design of image processing methods using optical flow [8], [9], [10] to proposals for motion estimation based on the subtraction of the background [11]. Chhaniyara *et al.* [8] focus on robotics platforms moving over different types of homogeneous terrains such as fine sand, coarse sand, gravel, etc. The rest of works [9], [10], [11] have been designed to estimate the speed of vehicles from video sequences acquired with a fixed video surveillance camera.

We, instead, propose to estimate the *proper* speed for a vehicle, according to the traffic situation, by using a vehicle *on-board* camera. While all the works mentioned above aim to estimate the actual speed at which the vehicle is moving, our ISA^2 models need to estimate the appropriate speed at which the vehicle should go. Our goal is not to know how fast a car goes, but how fast it should go.

III. ISA^2 DATASET

Here, we introduce the novel ISA^2 dataset, which allows us to train and test different approaches for the new challenging ISA^2 problem.

The database consists of 5 video sequences taken from both urban and interurban scenarios in the Community of Madrid, Spain. In total, we provide a set of 149.055 frames, with a size of 640×384 pixels, with the annotation of the proper speed of the car (km/h). During the driving for the acquisition of the dataset, in addition to respecting the speed limits, our driver has carefully tried to adjust the speed of the vehicle to what he considers to be an appropriate speed, according to the traffic situation. Figures 2(a) and 2(b) show some images of both, highway and urban routes, respectively.

To structure the database, both scenarios have been split into training and test subsets. For the 3 urban recordings, we use two of them for training/validation, an the third one for testing. We also provide two highway recordings, one for training/validation and the other for testing. These splits between training and testing have been done so that different scenarios and circumstances are well represented in both sets. Those scenarios include maximum and minimum speed over the sequences, stops at traffic lights or entrances and exists on the highway using service roads, for instance. Finally, with the aim of evaluating how well the different approaches are able to generalize, we introduce unique factors in the test subsets, such as, different weather conditions (rain) in the urban test set. All these aspects clearly help to release a challenging dataset. Table I shows the mean speed of the vehicle for the different subsets described.

TABLE I: Mean speed and standard deviation of the different sets in the ISA^2 dataset

Route	Set	Mean speed (km/h)	Std. deviation (km/h)
Highway	Training	84.31	18.15
Highway	Test	95.08	12.81
Urban	Training	19.55	13.60
Urban	Test	19.59	14.78

IV. MODELS FOR ISA^2

Our main objective during the design of the ISA^2 models is to propose a strong visual representation that allows the models to predict the appropriate speed for the vehicle.

The ISA^2 problem starts with a training set of images $S = \{(I_i, s_i)\}_{i=1}^{N}$, where N is the number of training samples. For each sample i in the dataset, I_i represents the input image, and $s_i \in \mathbb{R}$ encodes the annotation for the speed.

We first propose to learn a Convolutional Neural Network (CNN) [12] to directly perform the regression of the adequate speed. Technically, as it is shown in Figure 3, we use two different architectures: a VGG-16 [13] or a Residual CNN [14] (ResNet). Therefore, our networks are trained to learn the direct mapping from the image to the speed $\hat{s}$, a function that can be expressed as follows,

$$\hat{s}_W = f(W, I_i)\,, \tag{1}$$

where, $f(W, I_i) : I_i \rightarrow \mathbb{R}$ represents the mapping that the network performs to the input images. We encode in W the trainable weights of the deep architecture. We replace the loss function of the original network designs, which is no longer a softmax, but a loss based on the Euclidean distance.

(a) Highway (b) Urban

Fig. 2: Set of images from the ISA^2 dataset in highway and urban environments.

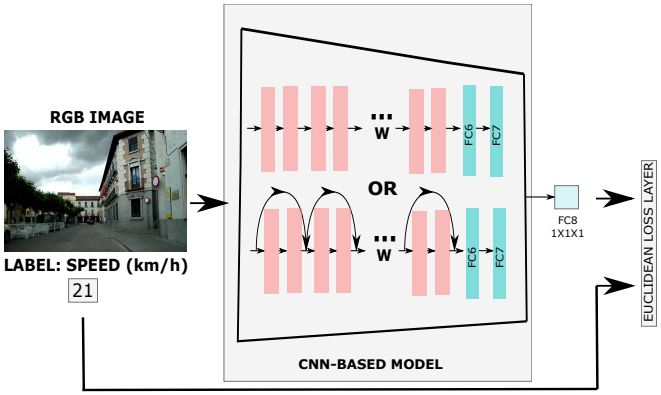


Fig. 3: ISA^2 from a CNN based architecture for regression.

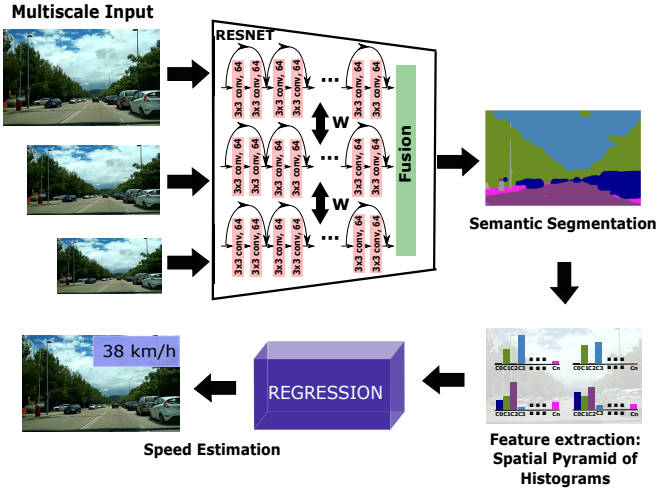


Fig. 4: ISA^2 from a semantic segmentation and a regressor.

The second approach is mainly based on a semantic segmentation model, see Figure 4. Our system starts performing a dense pixel labeling of the traffic scene. We then use a spatial pyramid pooling strategy, to build a descriptor for the image, which is based on the histogram of the different labels produced by our semantic segmentation model. This descriptor is used to learn a final regressor, which is the one in charge of the prediction of the proper speed.

Technically, for this second approach, we first implement the DeepLab [14] model, using a ResNet-101 as the base network. We train the DeepLab using a multi-scale input, using the scale factors $\{0.5, 0.75, 1\}$. We then fuse the prediction for each scale, taking the maximum response given by the network for each scale. Note that the ISA^2 dataset does not provide semantic segmentation annotations, therefore this model is trained using the Cityscapes dataset [15].

For the final regression, we evaluate in the experiments several approaches: linear regressor, lasso regressor, boosting trees and linear Support Vector Regressors. For all of them, we evaluate the impact of adding spatial information by using spatial pyramid pooling of up to 3 levels.

V. EXPERIMENTS

To evaluate the effectiveness of our models, we use here the ISA^2 dataset. We detail the experimental setup and main results in the following sections.

A. *Experimental setup and evaluation metric*

For our CNN-based approaches, VGG and ResNet-101, we fine-tune pre-trained models on the large-scale ImageNet dataset [16]. Both networks are trained for 4K iterations with a learning rate of 10^{-4} for the first 2K iterations, and of 10^{-5} for the rest. We use stochastic gradient descent (SGD) with a momentum of 0.9 and a batch size of 20 images for both architectures.

With respect to our models based on the semantic segmentation of the images, we cross validate both the specific parameters of the different regression methods and the spatial pyramid levels we use.

To measure the performance of the different models, we use the standard Mean Absolute Error (MAE) metric, which is defined as the difference in absolute value between the real

speed, s_r, and the proper speed estimated by an ISA^2 model, $\hat{s}$, averaged for the K images of the test set, according to:

$$\frac{1}{K}\sum_{i=1}^{K}|s_{r_i}-\hat{s}_i|. \quad (2)$$

We evaluate the MAE independently for the urban and highway set of images, because this provides a more detailed analysis of the results.

B. Quantitative results

In Table II we present the results of our ISA^2 approaches. In general, we show that our second approach, that is a semantic segmentation (SS) plus a regressor, obtains better results, only for the urban scenarios, than the first model proposed, where the CNNs directly cast the speed estimation. In a highway setting, our first approach reports a lower MAE. Probably, the fact that our first type of approaches have more parameters, allows them to adjust better the prediction to both types of environments.

TABLE II: MAE comparison of our different proposed methods. For each model, we train a unique regressor for both highway and urban scenarios.

Method	**Urban MAE** (Km/h)	**Highway MAE** (Km/h)
VGG-16	12.58	**11.57**
ResNet-101	11.49	11.87
SS + Linear regression	9.15	15.78
SS + SVR	10.69	16.76
SS + Lasso regression	**8.74**	18.13
SS + Boosting Trees	9.78	13.86

In this sense, we decide to perform a second experiment. We proceed to train an ISA^2 model for each type of scenario (urban and highway) separately. Table III shows the results. Now, models based on the SS perform better for both urban and highway images. In highway images, boosting trees are the ones that offer the best results, followed by the lasso regression and the SVR. On the other hand, in the urban sequences, a linear regression exhibits the best performance, followed by the lasso regression and the SVR. As a conclusion, it is clear that for our models based on SS, it is beneficial to train a regressor for each type of scenario separately. Figure 5 shows a graphical comparison of the results, following the two training methods described.

Finally, Figure 6 shows a graphical comparison between the proper speed of the vehicle (in blue) and the estimated speed (in red) by the different ISA^2 models proposed. For each type of scenario, results of the two CNN-based models used are shown together with the two best models based on SS + regression.

Interestingly, for the highway test sequence, all our models detect that it is necessary to reduce the speed halfway along the route, at a time when the driver leaves the highway towards a service road, to finally rejoin a different highway.

TABLE III: MAE comparison of our different proposed methods. For each model, we train an independent regressor for highway and urban scenarios.

Method	**Urban MAE** (Km/h)	**Highway MAE** (Km/h)
VGG-16	11.86	12.48
ResNet-101	9.59	12.79
SS + Linear regression	**6.02**	9.54
SS + SVR	8.14	9.23
SS + Lasso regression	6.67	8.72
SS + Boosting Trees	8.81	**7.76**

In general, we can observe that the neural networks have more difficultly to predict the proper speed, than the SS based solutions. This is particularly evident in the initial section of the routes, where the error made by the CNNs exceeds 30 km/h.

For the urban test sequence, it is remarkable that the CNNs are not capable of reducing the estimated proper speed when the vehicle is completely stopped, mainly at red traffic lights. On the other hand, SS-based regressors do adjust such situations much better.

C. Qualitative results

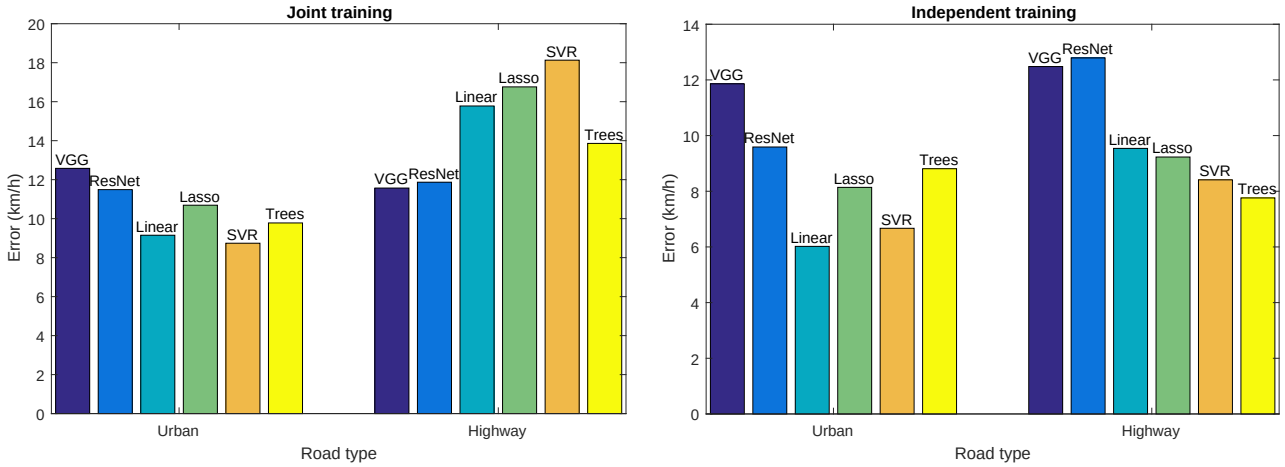
We show a set of qualitative results in Figure 7. Those results correspond to the best of our models for each type of road, i.e. using boosting trees in highway and SS + Linear regression in an urban environment.

Analyzing these results, we observe some of the difficulties our models have. On highways, for instance, the biggest errors for the estimation of the proper speed occur when the vehicle wants to leave the motorway, which leads the driver to slow down. Obviously, our models, which are based exclusively on what *the vehicle sees* at any given time, are not able to anticipate the driver's intentions, so they estimate a speed higher than the real one. However, as soon as the driver leaves the motorway and change the type of road, the models do correctly adjust the speed.

In urban environments, the main problem is related to the presence of stationary vehicles on the road, which implies that our vehicle has to stop when it reaches them. In those cases, although there is a decrease in the estimated proper speed, the models do not come to realize that it is necessary to completely stop. This does not occur in the presence of red traffic lights, where the estimated proper speed reaches 0 km/h.

VI. CONCLUSION

In this paper we propose for the first time the ISA^2 problem. It is a difficult and interesting problem, that has not been studied before. We also release a new dataset and propose an evaluation protocol to assist the research on ISA^2. Finally, we have introduced and evaluated two types ISA^2 models, and the results show the level of difficulty of the proposed task.



(a) Joint training

(b) Independent training

Fig. 5: MAE comparison between all of our different approaches to the ISA^2 problem.

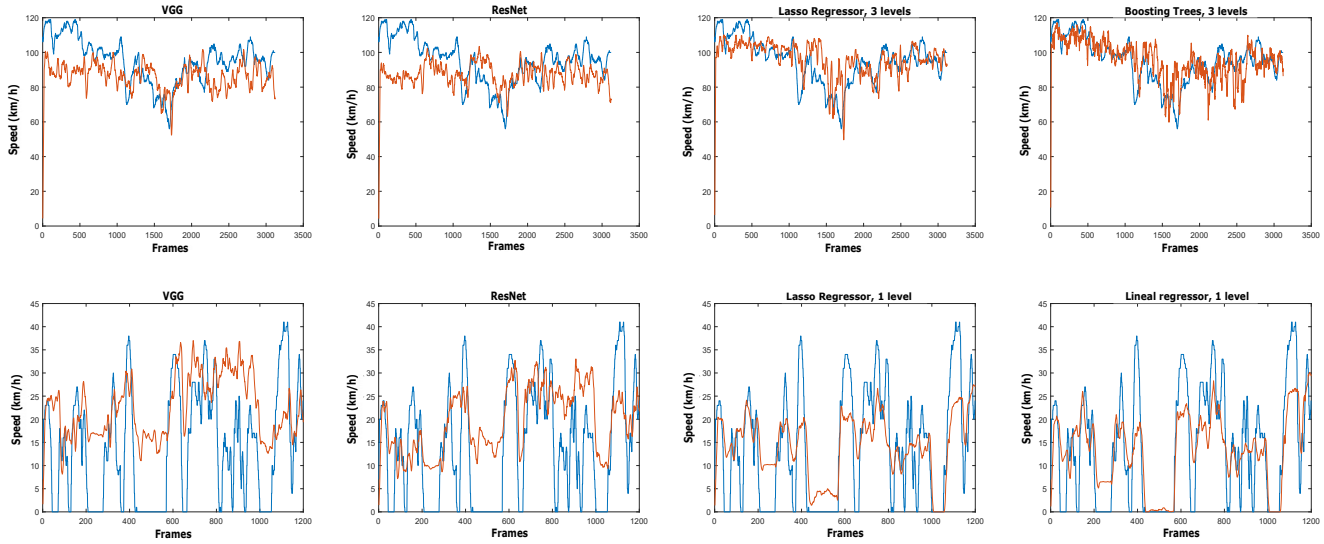


Fig. 6: Proper speed (blue) vs. Estimated proper speed (red) of different methods. First row corresponds to the highway test sequence, while second row shows results on the urban sequence.

The dataset and the proposed models will all be made publicly available to encourage much needed further research on this problem.

REFERENCES

[1] [Online]. Available: https://goo.gl/hKRUFn

[2] [Online]. Available: https://etsc.eu/12th-annual-road-safety-performance-index-pin-report/

[3] T. Vaa, T. Assum, and R. Elvik, "Driver support systems: Estimating road safety effects at varying levels of implementation," Institute of Transport Economics: Norwegian Centre for Transport Research, Tech. Rep. 1301/2014, March 2014.

[4] [Online]. Available: https://goo.gl/mAVvnJ

[5] D. Sierra-González, O. Erkent, V. Romero-Cano, J. Dibangoye, and C. Laugier, "Modeling driver behavior from demonstrations in dynamic environments using spatiotemporal lattices," in *ICRA*, 2018.

[6] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *CVPR*, 2017.

[7] F. Codevilla, M. Muller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *ICRA*, 2018.

[8] S. Chhaniyara, P. Bunnun, L. D Seneviratne, and K. Althoefer, "Optical flow algorithm for velocity estimation of ground vehicles: A feasibility study," *Journal On Smart Sensing And Intelligent Systems*, vol. 1, pp. 246–268, 01 2008.

[9] D. Shukla and E. Patel, "Speed determination of moving vehicles using lucas-kanade algorithm," *IJCATR*, vol. 2, pp. 32–36, 01 2012.

[10] I. Sreedevi, M. Gupta, and P. Asok Bhattacharyya, "Vehicle tracking and speed estimation using optical flow method," *International Journal of Engineering Science and Technology*, vol. 3, 01 2011.

[11] A. J. E. Atkociunas, R. Blake and M. Kazimianec, "Image processing in road traffic analysis," in *Image Processing in Road Traffic Analysis*, vol. 10, 2005, pp. 315–332.

[12] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *NIPS*, 1990, pp. 396–404.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

Fig. 7: Qualitative results of our best models for both type of roads. First two rows show a set of frames for which our ISA^2 solutions obtain the best predictions. Last row shows moments in which the speed difference is high.

[15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database." in *CVPR*, 2009.

Classification of Point Cloud for Road Scene Understanding with Multiscale Voxel Deep Network

Xavier Roynard[1] and Jean-Emmanuel Deschaud[1] and François Goulette[1]

***Abstract*— In this article we describe a new convolutional neural network (CNN) to classify 3D point clouds of urban scenes. Solutions are given to the problems encountered working on scene point clouds, and a network is described that allows for point classification using only the position of points in a multi-scale neighborhood. This network enables the classification of 3D point clouds of road scenes necessary for the creation of maps for autonomous vehicles such as HD-Maps.**

On the reduced-8 Semantic3D benchmark [1], this network, ranked second, beats the state of the art of point classification methods (those not using an additional regularization step as CRF). Our network has also been tested on a new dataset of labeled urban 3D point clouds for semantic segmentation.

I. INTRODUCTION

The majority of autonomous vehicles use 3D maps of the world for localization, perception and navigation tasks. As these maps improve the robustness of autonomous systems, we believe that almost all roads will be scanned in the future, representing for example 8 million km in North America and 4 million in Europe. Moreover, they must be updated regularly to take into account changes in the network. This is why it is important to set up the most automated processes possible to create and update these maps.

These point clouds must be processed after acquisition to extract the relevant information for autonomous driving: moving objects and parked vehicles must be removed, traffic signs, traffic lights and drivable areas detected. For example, for the localization task, the classified point cloud can be used as a map and moving objects can be detected with differences between the map and the current lidar frame as shown in figure 1.

To do so, the automatic classification of the data is necessary and is still challenging, regards to the number of objects present in an urban scene.

For the object classification task, deep-learning methods work very well on 2D images. The easiest way to transfer these methods to 3D is to use 3D grids. It works well when the data is just one single object [2].

But it is much more complicated for the task of point classification of a complete scene (e. g. an urban cloud) made up of many objects of very different sizes and potentially interwoven with each other (e. g. a lamppost passing through vegetation). Moreover, in this kind of scene, there are classes more represented (floor and buildings) than others (pedestrians, traffic signs ...).

This article proposes both a training method that balances the number of points per class during each epoch, and to our knowledge the first multi-scale 3D convolutional neural network applied to the semantic segmentation of 3D point clouds via multi-scale occupancy grids. These contributions significantly improve the state of the art of semantic segmentation methods without regularization of 3D point clouds of urban scenes.

[1]All authors are with Mines ParisTech, PSL Research University, Centre for Robotics. xavier.roynard@mines-paristech.fr

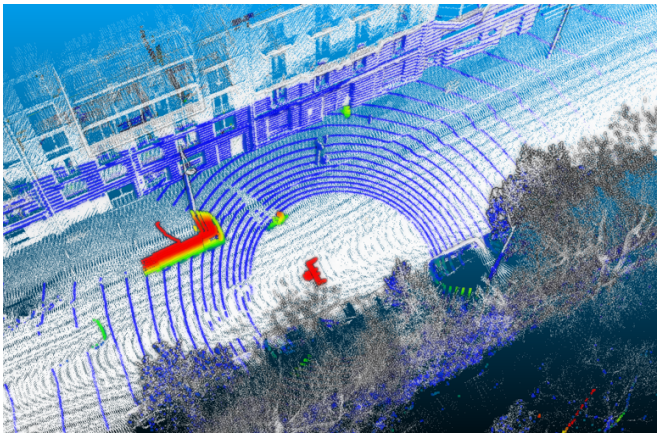


Fig. 1. Application of our classified point cloud for map based localization of an autonomous vehicle (in white, the map point cloud, in color the current velodyne frame of the autonomous vehicle from blue/close to the map to red/far to the map)

II. STATE OF THE ART

The focus here is on the semantic segmentation methods applied to dense registered point clouds used to create maps as HD-Maps, unlike the very sparse KITTI dataset clouds which require real-time processing methods.

A. Shallow and Multi-Scale Learning for 3D point cloud classification

There is a great variety of work for classifying 3D point cloud scenes by shallow learning methods or without learning. Methods can generally be classified into one of the two approaches: classify each point, then group them into objects, or conversely, divide the cloud into objects and classify each object.

The first approach is followed by [3] which classifies each point by calculating multi-scale features, computing the same kind of features at different scales to capture both context and local shape around the point. After classifying each point, the points can be grouped into objects by CRF [4] or by regularization methods [5].

The segmentation step of the second approach is usually heuristic-based and contains no learning. [6] segments the

cloud using super-voxels, [7] uses mathematical morphology operators and [8] makes a region growth to extract the soil, then groups the points by connected components. After segmentation, objects are classified by computing global descriptors that can be simple geometrical descriptors [7], or mixture of bag-of-words [9].

B. Deep-Learning for 3D point cloud classification

Over the past three years, there has been a growing body of work that attempts to adapt deep learning methods or introduces new "deep" approaches to classifying 3D point clouds.

This is well illustrated by the ShapeNet Core55 challenge [10], which involved 10 research teams and resulted in the design of new network architectures on both voxel grids and point cloud. The best architectures have beaten the state of the art on the two proposed tasks: part-level segmentation of 3D shapes and 3D reconstruction from single view image.

1) on 2D Views of the cloud:

The most direct approach is to apply 2D networks to images obtained from the point cloud. Among other things, we can think of the following projections:

- RGB image rendered from a virtual camera,
- depth-map, from a virtual camera,
- range image, directly from the sensor,
- panorama image[11],
- elevation-map.

These methods can be improved by taking multiple views of the same object or scene, and then voting or fusing the results [12] (ranked 5th on reduced-8 Semantic benchmark). In addition, these methods greatly benefit from existing 2D expertise and pre-trained networks on image datasets [13], [14] that contain much more data than point cloud datasets.

2) on Voxel Grid:

The first deep networks used to classify 3D point clouds date from 2015 with VoxNet [15], this network transforms an object instance by filling in an occupancy or density grid and then applies a Convolutional Neural Network (CNN). Later [16] applied the same type of network to classify urban point clouds, the network then predicts the class of a point from the occupancy grid of its neighborhood. However, we cannot compare with this architecture because the experimental data has not been published. Best results on ModelNet benchmarks are obtained using deeper CNNs [17] based on the architecture of Inception-ResNet [18] and voting on multiple 3D view of objects.

There are also significantly different approaches on voxel grids. OctNet [19] uses a hybrid Grid-Octree structure that allows CNNs to be used on resolved grids of higher reolution. VoxelNet [20] instead of increasing grid resolution, increases the size of voxels and the information contained in each voxel through a network similar to PointNet [21] (called Voxel Feature Encoding).

3) on Graph:

Another approach is to use graphs, indeed the raw point cloud having no structure, it is very difficult to derive general information from it. Whereas a graph gives relations of neighborhoods and distances between points and allows for example to make convolutions as in SPGraph [22] or to apply graph-cut methods on CRF as in SEGCloud [23].

4) on Point Cloud:

For the time being, there are still quite a few methods that take the point cloud directly as input. These methods have the advantage of working as close as possible to the raw data, so we can imagine that they will be the most efficient in the future. The first method of this type is PointNet [21] which gets fairly good results on ModelNet for object instance classification. PointNet is based on the observation that a point cloud is a set and therefore verifies some symmetries (point switching, point addition already in the set...) and is therefore based on the use of operators respecting these symmetries like the global Pooling, but these architectures lose the hierarchical aspect of the calculations that make the strength of the CNN. This gap has been filled with PointNet++ [24] which extracts neighborhoods in the cloud, applies PoinNet and groups the points hierarchically to gradually aggregate the information as in a CNN. Two other approaches are proposed by [25] to further account for the context. The first uses PointNet on multiscale neighborhoods, the second uses PointNet on clouds extracted from a 2D grid and uses recurrent networks to share information between grid boxes.

III. APPROACH

A. Learning on fully annotated registered point clouds

Training on scenes point cloud leads to some difficulties not faced when the point cloud is a single object. For the point classification task, each point is a sample, so the number of samples per class is very unbalanced (from thousands of points for the class "pedestrian" to tens of millions for the class "ground"). The classic training method by epoch would be to go through all the points of the training cloud at each epoch, making the classes with few samples anecdotal for the network.

We propose a training method that solves this problem. We randomly select N (for example $N = 1000$) points in each class, then we train on these points shuffled randomly between classes, and we repeat this process at the beginning of each Epoch.

Once a point p to classify is chosen, we compute a grid of voxels given to the convolutional network by building an occupancy grid centered on p whose empty voxels contain 0 and occupied voxels contain 1. We only use $n \times n \times n$ cubic grids where n is even, and we only use isotropic space discretization steps Δ. To reduce neighborhood search time, we can also sub-sample point clouds from the training set with a scale less than Δ.

B. Data Augmentation and Training

Some classic data augmentation steps are performed before projecting the 3D point clouds into the voxels grid:

- Flip x and y axis, with probability 0.5
- Random rotation around z-axis
- Random scale, between 95% and 105%

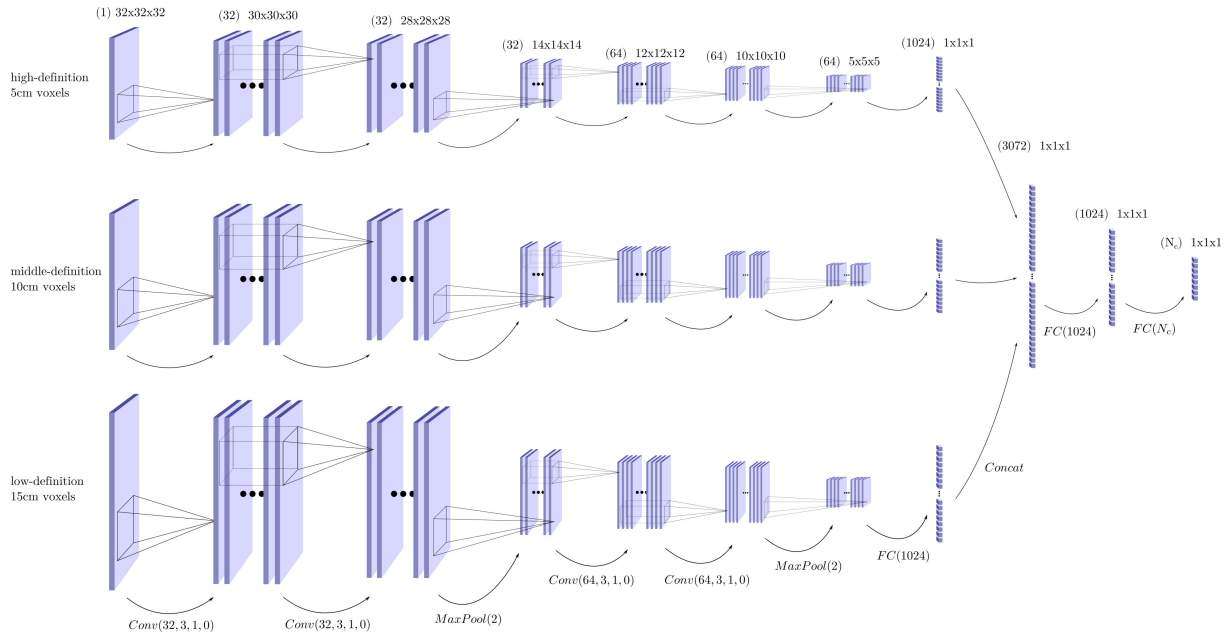


Fig. 2. Our Multi-Scale Voxel Network architecture: MS3_DeepVoxScene (all tensors are represented as 2D tensors instead of 3D for simplicity).

- Random occlusions (randomly removing points), up to 5%
- Random artefacts (randomly inserting points), up to 5%
- Random noise in position of points, the noise follows a normal distribution centered in 0 with standard deviation 0.01m

The cost function used is cross-entropy, and the optimizer used is ADAM [26] with a learning rate of 0.001 and $\varepsilon = 10^{-8}$, which are the default settings in most deep-learning libraries. To reduce neighborhood search time, we can also sub-sample point clouds from the training set with a scale less than Δ. In our experiments, all point clouds are subsampled at 2cm. No study has been carried out on the influence of subsampling on classification quality, but it is estimated that as long as the subsampling is performed at a scale below the discretization step of the voxel grid, the impact is negligible.

C. Test

To label a complete point cloud scene, the naive method is to go through all the points of the cloud, and for each point:

- look for all the neighboring points that fit into the occupation grid,
- create this grid,
- infer the class of the point via the pre-trained network.

However, two points very close to each other will have the same neighborhood occupancy grid and therefore the network will predict the same class. A faster test method is therefore to sub-sample the cloud to be tested. This has two beneficial effects: reduce the number of inferences and neighborhood searches, and each neighborhood search takes less time. To infer the point class of the initial cloud, we give each point the class of the nearest point in the subsampled cloud, which can be done efficiently if the subsampling method used retains the correct information.

IV. NETWORK ARCHITECTURE

The choosen network architecture is inspired from [28] that works well in 2D. Our network follows the architecture: $Conv(32,3,1,0) \rightarrow Conv(32,3,1,0) \rightarrow MaxPool(2) \rightarrow Conv(64,3,1,0) \rightarrow Conv(64,3,1,0) \rightarrow MaxPool(2) \rightarrow FC(1024) \rightarrow FC(N_c)$[1] where N_c is the number of classes, and each Convolutionnal ($Conv$) and Fully-Connected (FC) layer is followed by a Batch Normalization, a Parametric ReLU and a Squeeze-and-Excitation block [29] except the last FC layer that is followed by a $SoftMax$ layer. This network takes as input a 3D occupancy grid of size $32 \times 32 \times 32$, where each voxel of the grid contains 0 (empty) or 1 (occupied) and has a size of $10\text{cm} \times 10\text{cm} \times 10\text{cm}$.

This type of method is very dependent on the space discretization step Δ selected. Indeed, a small Δ allows to understand the object finely around the point and its texture (for example to differentiate the natural ground from the ground made by man) but a large Δ allows to understand the context of the object (for example if it is locally flat and horizontal around the point there can be ambiguity between the ground and the ceiling, but there is no more ambiguity if we add context).

Since a 3D scene contains objects at several scales, this type of network can have difficulty classifying certain objects. So we also propose a multiscale version of our network called MSK_DeepVoxScene for the K-scales version (or abbreviated in MSK_DVS).

[1] we denote $Conv(n,k,s,p)$ a convolutional layer that transforms feature maps from previous layer into n new feature maps, with a kernel of size $k \times k \times k$ and stride s and pads p on each side of the grid.

Name	LiDAR type	Covered Area	Number of points (subsampled)	Number of classes
Paris-Lille-3D [27]	multi-fiber MLS	55000m^2	143.1M (44.0M)	9
Semantic3D [1]	static LiDAR	110000m^2	1660M (79.5M)	8

TABLE I

COMPARISON OF 3D POINT CLOUD SCENES DATASETS. PARIS-LILLE-3D CONTAINS 50 CLASSES BUT FOR OUR EXPERIMENTATIONS WE KEEP ONLY 9 COARSER CLASSES. IN BRACKETS IS INDICATED THE NUMBER OF POINTS AFTER SUBSAMPLING AT 2 cm.

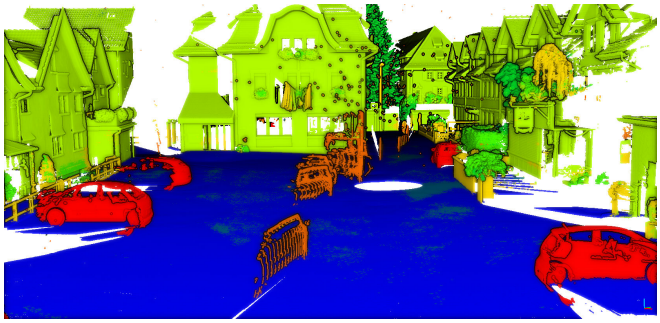


Fig. 3. Example of classified point cloud on Semantic3D test set (blue: man-made terrain, cerulean blue: natural terrain, green: high vegetation, light green: low vegetation, chartreuse green: buildings, yellow: hard scape, orange: scanning artefacts, red: cars).

We take several versions of the previous network without the fully-connected layer. The input of each version is given a grid of the same size $32 \times 32 \times 32$, but with different sizes of voxels (for example 5 cm, 10 cm and 15 cm). We then retrieve a vector of 1024 characteristics from each version, which we concatenate before giving to a fully-connected classifier layer. See figure 2 for a graphical representation of MS3_DeepVoxScene.

V. EXPERIMENTS

A. Datasets

To carry out our experiments we have chosen the 2 datasets of 3D scenes which seem to us the most relevant to train methods of deep-learning, Paris-Lille-3D [27] and Semantic3D [1]. Among the 3D point cloud scenes datasets, these are those with the most area covered and the most variability (see table I). The covered area is obtained by projecting each cloud on an horizontal plane in pixels of size 10cm $\times$ 10cm, then summing the area of all occupied pixels.

1) Paris-Lille-3D:

The Paris-Lille-3D dataset consists of 2 km of 3D point clouds acquired by Mobile Laser Scanning using with a Velodyne HDL-32e mounted on a van. Clouds are georeferenced using IMU and GPS-RTK only, no registration or SLAM methods are used, resulting in a slight noise. Because the scene is scanned at approximately constant speed, the point density is roughly uniform. The dataset consists of 3 files, one acquired in Paris and two acquired in Lille including `Lille1.ply` much larger than `Lille2.ply`. To validate our architectures by K-fold method, we cut spatially `Lille1.ply` into two folds containing the same number of points. Cross-validation is thus performed on 4 folds of similar sizes In addition, this dataset contains 50 classes, some of which only appear in some folds and with very few points. We therefore decide to delete and group together some classes to keep only 9 coarser classes:

ground	buildings	poles
bollards	trash cans	barriers
pedestrians	cars	natural

Some qualitative results on Paris-Lille-3D dataset are shown in figure 4. We can observe that some trunks of trees are classified as poles. It may means that the context is not sufficiently taken into account (even so the 15 cm grid is 4.8 m large) In addition, the ground around objects (except cars) is classified as belonging to the object. One can imagine that cars are not affected by this phenomenon because this class is very present in the dataset.

2) Semantic3D:

The Semantic3D dataset was acquired by static laser scanners, it is therefore more dense than a dataset acquired by MLS as Paris-Lille-3D, but the density of points varies considerably depending on the distance to the sensor. And there are occlusions due to the fact that sensors do not turn around the objects. Even by registering several clouds acquired from different viewpoints, there are still a lot of occlusions. To minimize the problem of very variable density, we subsample the training clouds at 2 cm. This results in a more uniform density at least close to the sensor and avoids redundant points. After subsampling, the dataset contains 79.5M points. The training set contains 15 point clouds which after sub-sampling are of similar sizes, each cloud is used as a separate fold for cross-validation. Some qualitative results on Semantic3D dataset are shown in Figure 3.

B. Evaluation Protocol

To confirm the interest of multi-scale CNNs, we compare the performance of our two architectures on these three datasets. And on Semantic3D we compare our results with those of the literature. The metrics used to evaluate performance are the following:

$$F1_c = \frac{2TP_c}{2TP_c + FP_c + FN_c}$$

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}$$

Where $F1_c$ and IoU_c represent respectively F1-score and Intersection-over-Union score of class c. And TP_c, TN_c, FP_c and FN_c are respectively the number of True-Positives, True-Negatives, False-Positives and False-Negatives in class c.

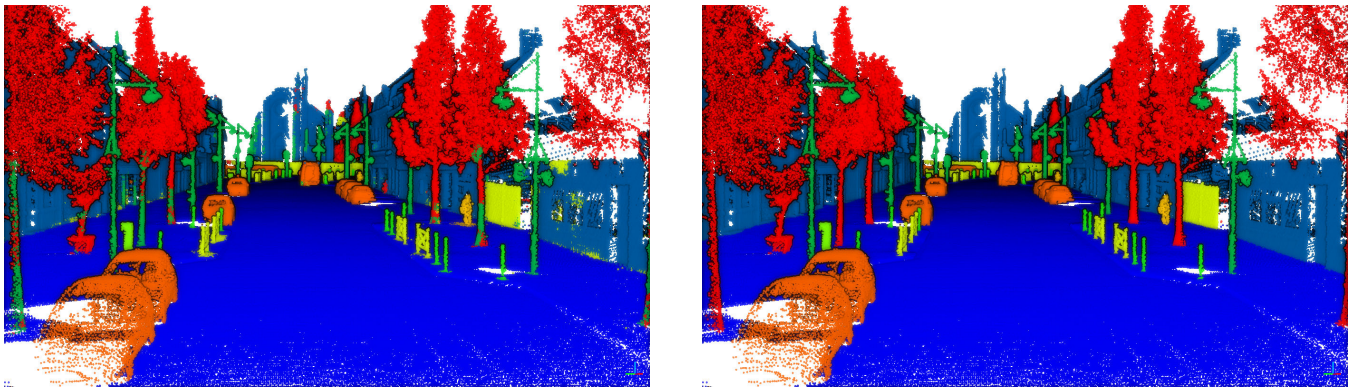


Fig. 4. Example of classified point cloud on Paris-Lille-3D dataset. Left: classified with MS3_DVS, right: ground truth (blue: ground, cerulean blue: buildings, dark green: poles, green: bollards, light green: trash cans, yellow: barriers, dark yellow: pedestrians, orange: cars, red: natural).

Rank	Method	Averaged IoU	Overall Accuracy	Per class IoU							
				man-made terrain	natural terrain	high vegetation	low vegetation	buildings	hard scape	scanning artefacts	cars
1	SPGraph[22]	**73.2**%	**94.0**%	**97.4**%	**92.6**%	**87.9**%	**44.0**%	**93.2**%	31.0%	**63.5**%	76.2%
2	MS3_DVS(Ours)	65.3%	88.4%	83.0%	67.2%	83.8%	36.7%	92.4%	**31.3**%	50.0%	**78.2**%
3	RF_MSSF	62.7%	90.3%	87.6%	80.3%	81.8%	36.4%	92.2%	24.1%	42.6%	56.6%
4	SegCloud[23]	61.3%	88.1%	83.9%	66.0%	86.0%	40.5%	91.1%	30.9%	27.5%	64.3%
5	SnapNet_[12]	59.1%	88.6%	82.0%	77.3%	79.7%	22.9%	91.1%	18.4%	37.3%	64.4%
9	MS1_DVS(Ours)	57.1%	84.8%	82.7%	53.1%	83.8%	28.7%	89.9%	23.6%	29.8%	65.0%

TABLE II

TOP-5 RESULTS ON SEMANTIC3D REDUCED-8 TESTING SET. MS3_DVS IS OUR MS3_DEEPVOXSCENE WITH VOXEL SIZES OF 5 cm, 10 cm AND 15 cm AND MS1_DVS IS OUR MS1_DEEPVOXSCENE WITH VOXEL SIZE OF 10 cm (ADDED FOR COMPARISON WITH NON MULTI-SCALE DEEP NETWORK).

Except for Semantic3D benchmark, all results are obtained by cross-validation by training on all folds except one and testing on the remaining fold. All our networks are trained for 100 epochs with 1000 points per class on each fold. No validation sets are used.

C. Comparison with the state of the art

For a comparison with the state-of-the-art methods on reduced-8 Semantic3D benchmark see table II. For MS1_DeepVoxScene several resolutions have been tested, and by cross-validation on the Semantic3D training set the 10 cm resolution is the one that maximizes validation accuracy. DeepVoxScene's choice of MS3_DeepVoxScene resolution results from this observation, we keep a resolution that obtains good performance in general, and we add a finer resolution of 5 cm to better capture the local surface near the point, and a coarser resolution of 15 cm to better understand the context of the object to which the point belongs. Our method achieves better results than all methods that classify cloud by points (i. e. without regularization).The inference time of the 23.5 million points of the reduced8 test set subsampled at 2 cm is approximately 32 h. And the propagation of classes to the nearest points on the original cloud (not subsampled) takes approximately an hour.

Dataset \ Method	MS3_DVS	MS1_DVS	VoxNet [15]
Paris-Lille-3D	**89.29**%	88.23%	86.59%
Semantic3D	**79.36**%	74.05%	71.66%

TABLE III

COMPARISON OF MEAN F1 SCORES OF MS3_DVS, MS1_DVS AND VOXNET [15]. FOR EACH DATASET, THE F1 SCORE IS AVERAGED ON ALL FOLDS.

D. Study of the different architectures

To evaluate our architecture choices, we tested this classification task by one of the first 3D convolutional networks: VoxNet [15]. This allows us both to validate the choices made for the generic architecture of the MS1_DeepVoxScene network and to validate the interest of the multi-scale network. We reimplemented VoxNet using the deep-learning library Pytorch. See table III for a comparison between VoxNet [15], MS1_DeepVoxScene and MS3_DeepVoxScene on the 3 datasets.

See table IV for a comparison per class between MS1_DeepVoxScene and MS3_DeepVoxScene on Paris-Lille-3D dataset. This shows that the use of multi-scale networks improves the results on some classes, in particular

Class	Precision		Recall	
	MS3_DVS	MS1_DVS	MS3_DVS	MS1_DVS
ground	**97.74**%	97.08%	**98.70**%	98.28%
buildings	**85.50**%	84.28%	**95.27**%	90.65%
poles	**93.30**%	92.27%	92.69%	**94.16**%
bollards	98.60%	**98.61**%	93.93%	**94.16**%
trash cans	**95.31**%	93.52%	79.60%	**80.91**%
barriers	**85.70**%	81.56%	**77.08**%	73.85%
pedestrians	**98.53**%	93.62%	**95.42**%	92.89%
cars	93.51%	**96.41**%	**98.38**%	97.71%
natural	**89.51**%	88.23%	**92.52**%	91.53%

TABLE IV
PER CLASS PRECISION AND RECALL AVERAGED ON THE 4 FOLDS OF PARIS-LILLE-3D DATASET.

the buildings, barriers and pedestrians classes are greatly improved (especially in Recall), while the car class loses a lot of Precision.

VI. CONCLUSIONS

We have proposed both a training method that balances the number of points per class seen during each epoch, as well as a multi-scale CNN that is capable of learning to classify point cloud scenes. This is achieved by both focusing on the local shape of the object around a point and by taking into account the context of the object in a multi-scale fashion.

We validated the use of our multi-scale network for 3D scene classification by ranking second on Semantic3D benchmark and by ranking significantly better than state-of-the-art point classification methods (those without regularization).

REFERENCES

[1] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d.net: A new large-scale point cloud cassification benchmark," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98.

[2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[3] T. Hackel, J. D. Wegner, and K. Schindler, "Fast semantic segmentation of 3d point clouds with strongly varying density," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Prague, Czech Republic*, vol. 3, pp. 177–184, 2016.

[4] R. Zhang, S. A. Candra, K. Vetter, and A. Zakhor, "Sensor fusion for semantic segmentation of urban scenes," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1850–1857.

[5] L. Landrieu, H. Raguet, B. Vallet, C. Mallet, and M. Weinmann, "A structured regularization framework for spatially smoothing semantic labelings of 3d point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 132, pp. 102 – 118, 2017.

[6] A. K. Aijazi, P. Checchin, and L. Trassoudaine, "Segmentation based classification of 3d urban point clouds: A super-voxel based approach with evaluation," *Remote Sensing*, vol. 5, no. 4, pp. 1624–1650, 2013.

[7] A. Serna and B. Marcotegui, "Detection, segmentation and classification of 3d urban objects using mathematical morphology and supervised learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 93, pp. 243–255, 2014.

[8] X. Roynard, J.-E. Deschaud, and F. Goulette, "Fast and robust segmentation and classification for change detection in urban point clouds," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B3, pp. 693–699, 2016.

[9] J. Behley, V. Steinhage, and A. B. Cremers, "Laser-based segment classification using a mixture of bag-of-words," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, nov 2013, pp. 4195–4200.

[10] L. Yi, H. Su, L. Shao, M. Savva, H. Huang, Y. Zhou, B. Graham, M. Engelcke, R. Klokov, V. Lempitsky, *et al.*, "Large-scale 3d shape reconstruction and segmentation from shapenet core55," *arXiv preprint arXiv:1710.06104*, 2017.

[11] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval," *Computers & Graphics*, 2017.

[12] A. Boulch, B. L. Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Eurographics Workshop on 3D Object Retrieval*, vol. 2, 2017, p. 1.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[15] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[16] J. Huang and S. You, "Point cloud labeling using 3d convolutional neural network," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 2670–2675.

[17] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *arXiv preprint arXiv:1608.04236*, 2016.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[19] G. Riegler, A. O. Ulusoys, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," *arXiv preprint arXiv:1611.05009*, 2016.

[20] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2016.

[22] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," *arXiv preprint arXiv:1711.09869*, Nov. 2017.

[23] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," *arXiv preprint arXiv:1710.07563*, 2017.

[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5105–5114.

[25] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 716–724.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: a large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification," *ArXiv e-prints*, Nov. 2017.

[28] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv e-prints*, Sept. 2014.

[29] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *ArXiv e-prints*, Sept. 2017.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Navigation, Decision, Safety

- **Keynote speaker: Maucher Dominik (Bosch, Germany)**
  Title: Fully Automated Driving: Results and Open Challenges in Intelligent Decision Making, Planning, and Maps

- **Title: Statistical Model Checking Applied on Perception and Decision-making Systems for Autonomous Driving**
  Authors: J. Quilbeuf, M. Barbier, L. Rummelhard, C. Laugier, A. Legay, B. Baudouin, T. Genevois, J. Ibanez-Guzman, and O. Simonin

- **Title: Automatically Learning Driver Behaviors for Safe Autonomous Vehicle Navigation**
  Authors: E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Keynote speaker: **Maucher Dominik**
(Bosch, Germany)

Fully Automated Driving: Results and Open Challenges in Intelligent Decision Making, Planning, and Maps

Abstract: The current Space Race of autonomous driving is in full swing. We will attempt to paint a picture of how far we have gotten and what are some of the most pressing current challenges. We will touch upon some of the unanswered questions in the areas of intelligent decision making, planning, and maps. We will also give you a glimpse into the joint Bosch-Mercedes automated driving project, which aims to bring autonomous cars to the roads at the beginning of the next decade.

Biography:

Dominik Maucher is a research engineer for Behavioral Planning at Robert Bosch GmbH, Stuttgart. He received is M.Sc. in Computer Science from the Karlsruhe Institute of Technology in 2014. He started with Perception and indoor Localization for warehouse Robotics by introducing novel research SLAM to real-time and low cost systems. In 2014 he has started working at Robert Bosch GmbH on Robotics and Automated Driving in various domains including Perception and sensor data fusion for highly Automated Driving on highways, as well as behavior planning for highly and fully automated driving. Since 2017, he has been working in the Bosch/Daimler cooperation on urban automated driving. His current fields of activity are scene-understanding, Behavioral Planning and Prediction.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Navigation, Decision, Safety

- **Title: Statistical Model Checking Applied on Perception and Decision-making Systems for Autonomous Driving**
  Authors: J. Quilbeuf, M. Barbier, L. Rummelhard, C. Laugier, A. Legay, B. Baudouin, T. Genevois, J. Ibanez-Guzman, and O. Simonin

- **Title: Automatically Learning Driver Behaviors for Safe Autonomous Vehicle Navigation**
  Authors: E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Statistical Model Checking Applied on Perception and Decision-making Systems for Autonomous Driving.

Jean Quilbeuf[1], Mathieu Barbier[2,3], Lukas Rummelhard[4], Christian Laugier[2],
Axel Legay[1], Blanche Baudouin[2], Thomas Genevois[2], Javier Ibañez-Guzmán[3], and Olivier Simonin[2,5]

[1]Univ Rennes, Inria, F-35000, RENNES, France., Email: name.surname@inria.fr
[2]Univ. Grenoble Alpes, Inria, Chroma, F38000 Grenoble. Email: name.surname@inria.fr
[3]Renault S.A.S, 1 av. du Golf, 78288 Guyancourt, France. Email: name.surname@renault.com
[4]Univ. Grenoble Alpes, CEA, LETI, DSYS, LSOSP, F38000 Grenoble.
[5]INSA Lyon, CITI Lab., 6 avenue des Arts, 69680 Villeurbanne, France. Email: name.surname@inria.fr

***Abstract*—Automotive systems must undergo a strict process of validation before their release on commercial vehicles. The currently-used methods are not adapted to latest autonomous systems, which increasingly use probabilistic approaches. Furthermore, real life validation, when even possible, often imply costs which can be obstructive. New methods for validation and testing are necessary.**

In this paper, we propose a generic method to evaluate complex automotive-oriented systems for automation (perception, decision-making, etc.). The method is based on Statistical Model Checking (SMC), using specifically defined Key Performance Indicators (KPIs), as temporal properties depending on a set of identified metrics. By feeding the values of these metrics during a large number of simulations, and the properties representing the KPIs to our statistical model checker, we evaluate the probability to meet the KPIs. We applied this method to two different subsystems of an autonomous vehicles: a perception system (CMCDOT framework) and a decision-making system. An overview of the two system is given to understand related validation challenges. We show that the methodology is suited to efficiently evaluate some critical properties of automotive systems, but also their limitations.

I. Introduction

In the automotive industry the development and testing of human centric-systems must follow the guidelines of the ISO26262. In the automotive industry, this kind of testing can be divided in two:

- Vehicle-in-the-loop platform tests interactions between a human and the system in dangerous situation [1].
- Hardware-in-the-loop to test interactions between an embedded system, such as the Active Brake Control Systems [2], and the physics of a vehicle.

For autonomous functionality higher than the level 3 as define by the SAE, drivers will not be responsible of most of driving decisions. As these systems will rely on machine learning and probabilistic methods, conventional methods for validation are not adapted. The vehicle shall operate in a various range of scenarios as well as dangerous situations, the validation and verification operations must be carried on simulations perform. It allows to reduce cost and increase the coverage of system testing.

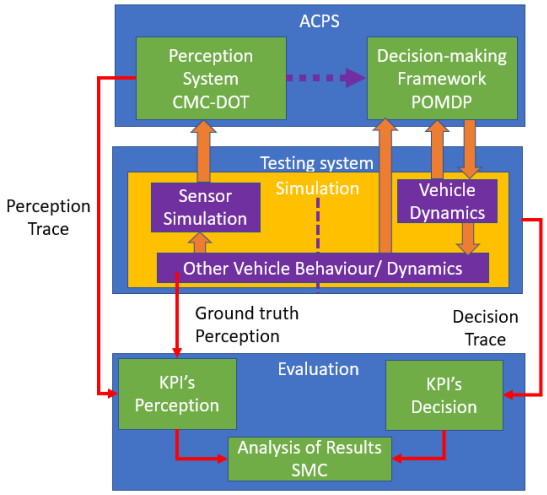


Figure 1. Interactions between the different elements of the proposed Validation pipeline. Dashed line represents future developments to connect the decision-and perception

The difficulty for the validation of an autonomous system are two-fold. First, the complexity and variety scenarios that autonomous vehicle will face is larger than in Advanced driver assistance systems (ADAS). Second, multiple systems will be in constant interaction. In this study we focus on a use-case that highlights these two difficulties: road intersection crossing. It is one of the most dangerous part of the road network with more than 8% of the total road fatalities in Europe [3]. Furthermore, there exists many variations of each scenario (number of vehicles, initial velocities, etc.). It is challenging for the perception because of the limited view range, partially-observed vehicles and because of the presence of multiple vulnerable users that could be potentially at risk. For the decision-making, the interactions between road users

are complex to consider because of wrong behaviour of other drivers. Road intersection crossing has been identified as one use-case addressed in the Enable-3 European project [4]. This industry-driven project aspire to propose methods for validation and verification of automated cyber-physical systems (ACPS). The global architecture for validation and verification has been simplified to match our thematic and is illustrated in figure 1.

Examples of validation for highly-autonomous systems can be found in the aerospace domain [5], where formal methods are used to validate the behavior of a fleet of satellites. In the robotic domain, benchmarks allow researchers to compare their results in the same conditions [6], [7]. However benchmarks are often tailored for one specific kind of problem and are not representative enough of the variety of situation that an autonomous system may encounter to actually validate such a system. Waymo was recently confident enough in their system to remove the safety drivers for some tests. This was possible with an effort of 1 billion kilometers driven in a simulated environment [8]. Another way is to use formal methods to ensure the safety of the vehicle [9] but it would rather complex to do in uncertain environment.

The purpose of the paper is to demonstrate the use of a validation method (SMC) on two different systems that have been previously developed, namely Perception and Decision. The requirements for the testing in simulated environment are discussed for each system. Preliminary results for the decision-making system are presented as well as discussions on the challenges caused by the perception system.

Section II presents our validation approach based on statistical model checking. Section III describes the application of our approach on the perception system and the difficulty to find applicable metrics for its validation. Then Section IV shows a more complete application and interprets the results for the decision-making system.

II. Statistical Model Checking

In the context of ACPS, it is not possible to afford validation through exhaustive techniques, that is by stating a property and checking that it holds in all reachable states. Indeed, this would require to model and traverse all the reachable states of the ACPS. Such a modelling is possible at a very abstract level, but requires a huge effort to be brought at a more detailed level. Furthermore, even if a very detailed model of the ACPS were provided, exploring all its reachable states would not be possible due to the very large state space. Stochastic algorithm are complex to validate with conventional methods, thus it is interesting to use probabilistic methods to evaluate them [10].

Statistical Model Checking (SMC) [11], [12] provides an intermediate between test and exhaustive verification by relying on statistics. In order to perform SMC, one needs an executable model and a property to check. The executable model is expected to be stochastic, that is, to have some of its transition governed by probabilistic choices. Note that most ACPS simulations are already modelled as stochastic processes, because variations in the scenario are defined by probability distributions. The property to check must be decidable on a finite trace.

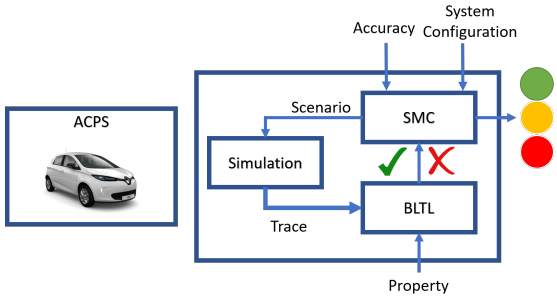


Figure 2. An overview of SMC

The execution being stochastic, some traces will satisfy the property to check and some other will not. Therefore, we can define the probability that a trace satisfies a property. The main goal of SMC is to evaluate that probability. Note that a probability of satisfying a formula gives actually more information than a yes-or-no answer. Indeed, if the model does not satisfy the formula, there is an evaluation of how well it performs.

In order to perform SMC, one needs to be able to

- Generate traces of the execution of the system to validate. These traces have to be generated according to the probabilities in the model.
- Write the property to check as a formula that can be decided on a finite trace, and a procedure for deciding whether a trace satisfy the property.

We present in Figure 2 an overview of the approach. On the left we have a simulator that provides stochastic execution of our system. On the bottom we have the property φ to check. On the top, we have some configuration for the SMC algorithm, such as the required accuracy. The SMC algorithm requires some simulations to the simulator. In turn the simulator provides a trace σ that is fed to the property checker. Finally the property checker returns its verdict to the SMC algorithm. At this point, if the SMC algorithm has enough information to return a result that meets the required accuracy, it does so. Otherwise, it asks for an additional simulation and the loop is run again. We give an intuition of SMC by illustrating it with the Monte-Carlo Algorithm. This algorithm estimates the probability p that a system satisfies a property P by checking P against a set of N random executions of the system. The estimation $\hat{p}$ is given by

$$\hat{p} = \frac{1}{N}\sum_{1}^{N} f(ex_i) \quad where f(ex_i) = \begin{cases} 1 & \text{if} ex_i \models P \\ 0 & \text{otherwise} \end{cases}$$

Using the formal semantics of the property language, the property is checked against each execution trace. The trace must be long enough to decide whether the property holds.

Of course, the larger is the set of simulations, the more precise is the result. The confidence bounds of the estimation

are set by two positive real parameters ϵ and δ. The confidence is defined by the Chernoff bound that is stated as:

$$Pr(|p - \hat{p}| \leq \epsilon) \geq 1 - \delta$$

Assuming that p is value of the probability we want to evaluate and $\hat{p}$ is the estimation we compute, the formula means that the estimation error, i.e. the distance $|p - \hat{p}|$, is bound by ϵ with a probability $1 - \delta$. In other words, the probability that the error in the estimation is greater that ϵ is δ. Once δ and ϵ have been set, we can compute the number of simulations N necessary to enforce the above formula. The quality of the approximation is high (and thus N is high as well) when ϵ and δ are close to 0. When ϵ and δ increase, the estimation is more approximate but requires less simulations to be computed.

A. Defining KPIs

In order to define and evaluate KPIs based on a set of simulations, we proceed as follows. We first identify with peoples in charge of developing the system some KPIs related to system under test and scenarios. We then express the KPIs as temporal formulas involving the identified metrics. Temporal formulas allow a finer formulation of KPIs by taking into account the evolution of the metrics during time. Let us consider acceleration as a metric. A rough formulation of a KPI concerning acceleration might be that the acceleration should be bounded, i.e. to guarantee the comfort of the passengers [13]. A finer formulation could be that the acceleration should generally be bounded, but the bound can be exceeded for a short period of time.

In order to express such formulas, we rely on BLTL, a bounded version of LTL [14]. The syntax of BLTL is as follows: $\phi \ ::= p \mid \phi \vee \phi \mid \neg\phi \mid \phi U_{\leq t}\, \phi \mid X_{\leq t}\, \phi$. A BLTL formula is expressed with respect to a trace. In our case a state is a sequence of states, one for each simulation step. Each state contains the value of each of the metrics at that current state. The symbol p represents a predicate expressed on the current state, for instance a comparison between a metric and a bound. The disjunction ($\vee$) and the negation ($\neg$) defined as usual. Finally, the temporal operators until (U) and next (X) define properties about the time. Since we need to be able to decide whether a property holds on a finite trace, these operators are parameterized by a time bound $t \in \mathbb{R}$. The formula $X_{\leq t}\phi$ is true if ϕ is true in the state reached after t units of time from the current state. The formula $\phi_1\, U_{\leq t}\, \phi_2$ is true if 1) the formula ϕ_2 becomes true before t units of time from the current state and 2) the formula ϕ_1 remains true in every state before the one where ϕ_2 becomes true. For a formal definition of BLTL semantics, see [15].

In practice, we often use the *always* (G) and *eventually* (F) operators. Eventually is defined as $F_{\leq t}\phi = \texttt{true}\, U_{\leq t}\, \phi$ and means that the formula ϕ should become true before t units of time happen. Always is defined as $G_{\leq t}\phi = \neg F_{\leq t}\neg\phi$ and means that ϕ must always hold for the next t units of time.

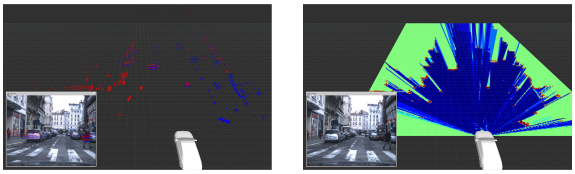


Figure 3. Data fusion in an occupancy grid. Data from each of the 2 LiDARs are used to generate occupancy grids using sensor models, which are then fused by Bayesian fusion.

III. A first validation application: CMCDOT Perception system

A. Principle of the CMCDOT

The CMCDOT Framework is a perception system, based on environment representation through probabilistic occupancy grids, a dense and generic representation [16], [17], and Bayesian fusion, filtering and inference.

This type of Bayesian formalism [18] allows proper confidence estimation and combination, particularly important features when confronted with incomplete or even contradictory data coming from different sensors. A major feature of the system is its highly-parallelized design: from data fusion, to grid filtering, velocity inference and collision risk assessment, the methods have been designed to allow massive parallelization of computations, and so benefit from parallel-computing devices [19], allowing real-time performances on embedded devices.

Sensor data is converted to occupancy estimation using specific sensor model, sensor occupancy estimates are then combined by Bayesian fusion in every grid cell (Fig. 3). The Conditional Monte Carlo Dense Occupancy Tracker (CMCDOT) [20] itself is a generic spatial occupancy tracker, which then infers dynamics of the scene through a hybrid representation of the environment consisting of static and dynamic occupancy, empty spaces and unknown areas(Fig. 4). This differentiation enables the use of state-specific models (classic occupancy grids for motionless components and sets of moving particles for dynamic occupancy), as well as relevant confidence estimation and management of data-less areas. The approach leads to a compact model that dramatically improves the accuracy of the results and the global efficiency in comparison to previous approaches.

This method is particularly suitable for heterogeneous sensor data fusion (camera, lidars, radars etc…). The occupancy of each cell over time can be estimated from various sensors data whose specific uncertainty (noise, measurement errors) are taken into consideration. Filtered cell estimates are thus much more robust, leading to a more reliable global occupancy of the environment, reducing false detections.

While most of risk estimation methods consist in detecting and tracking dynamic objects in the scene [21], [22], the risk being then estimated through a Time to Collision (TTC) approach by projecting object trajectories to the future [23], [24], the grid-based approach used in the CMCDOT framework[20] instead directly computes estimations of the

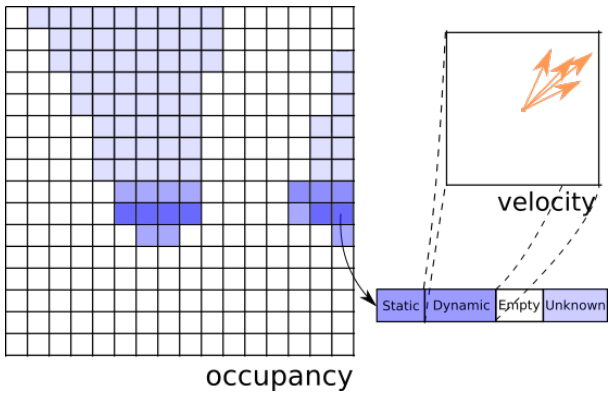


Figure 4. Data representation in the CMCDOT formulation. The environment is divided into cells, to which are associated static, dynamic, empty and unknown coefficients. The dynamic part is allotted to weighted particles which sample the velocity space

position in the near future of every static and dynamic part of the grid, as well as the trajectory of the vehicle. These estimations are iteratively computed over short time periods, until a potential collision is detected, in which case a TTC is associated to the cell from which the colliding element came from (Fig. 5). In every cell, the associated TTCs are cumulated over different time periods (1, 2, 3 seconds for example) to estimate a cell-specific collision risk profile. Risk grids, and global aggregated risks, are thus generated, and later used to generate response impulses for the control system. This strategy[25] avoids solving the complex problem of multi-object detection and tracking, while integrating the totality of the available information. It provides a probabilistic estimation of the risk associated to each part of the scene.

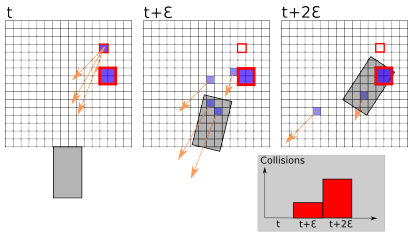


Figure 5. Collision risk estimation over time for a specific cell. The cell position is predicted according to its velocity, along with the mobile robot. This risk profile is computed for every cell, and then used to integrate over time the global collision risk.

B. Method Application

1) Simulation for perception: In this project, the simulation relies on the use of two frameworks: Gazebo and ROS. Gazebo allows for the representation and simulation of the environment, the ego vehicle and its sensors, as depicted in Figure 6. Each item in these three categories is matched with a visual representation and physical characteristics (dimensions, weight, friction, etc). The data acquisition and processing part of the simulation is carried out in ROS, where the data can be recorded, stored, and processed by the same code running on

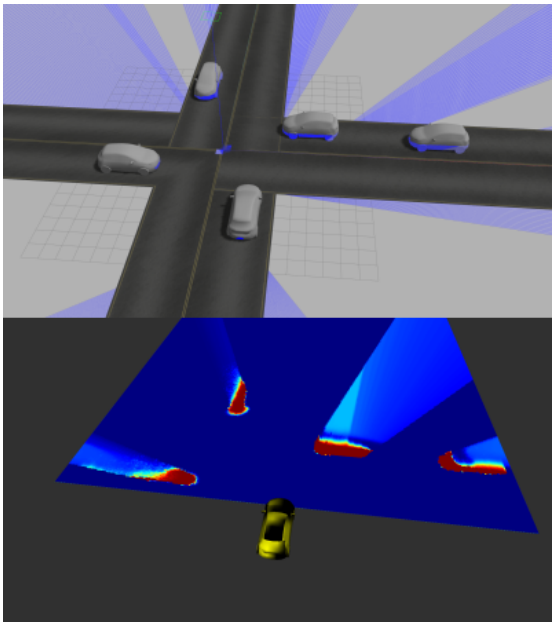


Figure 6. Simulated scenario for the CMCDOT algorithm (top),Output of CMCDOT (bottom)

the actual vehicle. The communication between the ROS and Gazebo modules is carried out seamlessly thanks to the native use of ROS messages. In order for our simulation approach to be precise and fully exploitable, the simulation framework must provide the following elements:

- precise volume and shape of each vehicle, and surface reflectivity.
- atmospheric conditions which might impact the vehicles' trajectory (wind gusts) or lidar detection (heavy rain or snow).
- in order to establish the ground truth, a grid indicating the position of all simulated objects. This grid must reflect CMCDOT's occupation grid in the following aspects: origin position, grid direction, cell size.

Currently, each lidar is simulated with the appropriate position on the ego vehicle, the same sampling frequency and the same data format as the physical sensor. To match the sensing uncertainty, a Gaussian noise can be added.

In order to be able to efficiently generate a large number of simulated environments, we have perfected a parameter-based approach which streamlines the process through which the dimensions and initial position and velocity of non-ego vehicles are specified.

Our simulation scenario aims at checking the behaviour of cars at a four-way crossroads. The rule governing this crossroad is that at any given moment in time, a maximum of one simulated vehicle is present on the crossroad. To simulate the different cases, we rely on the random generation of parameter sets (non-ego vehicle class, initial position and initial speed). The test cases are then run, and their results (perception results as in Figure 6) are stored alongside the parameter sets. The analysis of these datasets enables us to

accurately measure the efficiency of our perception and control solution.

The strong advantage of this approach is the ease with which a large number of simulated scenarios can be generated, ran, and analyzed.

2) KPI definition: Contrary to most perception systems, outputs of CMCDOT are not a direct list of detected objects, but dynamic occupancy grid, a rich probabilistic representation of the entire surrounding space. While object detector metrics are already not perfectly defined, the topic of evaluation of occupancy grids (furthermore dynamic occupancy grids, incorporating at a cell level velocity field estimations) is an important subject [26].

A first approach is to define a global indicator based on the direct estimates of the grid, in comparison to the ground truth. But if by qualitative analysis of results it is quite simple to evaluate if an occupancy grid is correct or not, an objective quantification of this quality is particularly complicated, each metrics focusing on a specific aspect, ignoring others (for example occupied / free space factor, cell by cell comparison, convolution-based metrics, etc.).

Another approach is to focus on specific applications of the method: the validation of the whole system itself is performed by statistical validation of its usages. In the case of the CMCDOT framework, a direct application of the perception system is an automatic braking system, based on aggregated risk estimates of the system. By comparing the difference in response of the system and expected behavior according to the ground truth, a partial evaluation of the system can be accessed.

In order to assess the correctness of the CMCDOT algorithm, we compare the output of the algorithm to the actual context of the car in the simulation. We focus on the risk of collision at 1, 2 and 3 seconds.

In order to evaluate the correctness of this output, we extract a traces of the simulation containing the following metrics: $\texttt{cmcdot\_risk}_i$ and $\texttt{real\_coll}_i$ for $1 \leq i \leq 3$. The metric $\texttt{cmcdot\_risk}_i$ indicate the probability of a collision in i s according to the CMCDOT algorithm. The metric $\texttt{real\_coll}_i$ is a Boolean indicating whether a collision will occur if object continue to move with their current speed, according to their speed and position in the simulation.

We define one KPI for each time interval, parameterized by a threshold τ. We formalize our KPI through the property $G_{\leq t}(\texttt{real\_coll}_i \Rightarrow (1 - \texttt{cmcdot\_risk}) < \tau) \wedge (\neg\texttt{real\_coll}_i \Rightarrow \texttt{cmcdot\_risk}) < \tau)$. This property states that if there is a risk of collision, the probability returned by CMCDOT must be high enough. Conversely, if there is no risk of collision, the probability returned by CMCDOT must be small enough.

IV. A second validation application: a decision-making system

A. Principle of the POMDP based decision-making

The decision-making system is a key component of an autonomous vehicle. Its task is to plan the movement of the vehicle taking into account the uncertainty in the situation measurement as well as the uncertain consequences of its action will have on the situation.

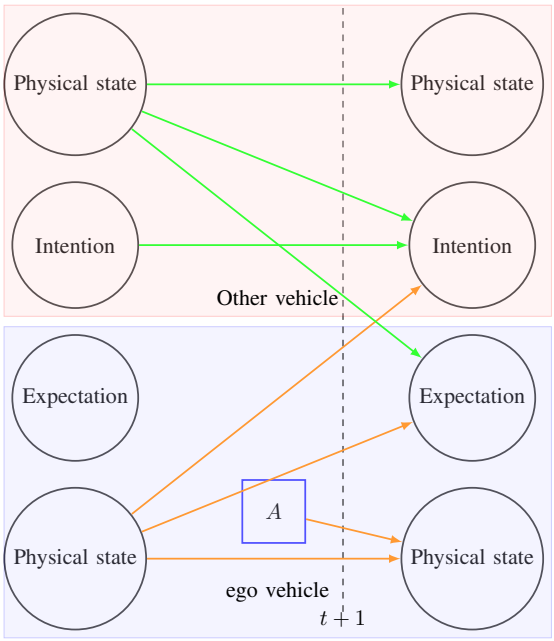


Figure 7. The POMDP represented as Bayesian network. The square node represent the action chosen by the framework

Partially observable Markov decision process (POMDP) is a mathematical model that formalizes this two kind of uncertainties and has been used for planning in stochastic environment [27].

With recent advancement on online-Pomdp solver [28](used in our work), complex problems such as road intersection crossing has been addressed in [29]. The key element of our approach [30] is to take into account the difference between intention and expectation of drivers approaching an intersection (inspired from [31]) to enable partial cooperation. The intention corresponds to the manoeuvre actually performed by the drivers and could observed with the approach developed in [32]. The expectation represent what the driver should do regarding the current situation and traffic rules. Situation where intention and expectation does not match could result in risky interactions. These two variables can be inferred from the physical state (Velocity and distance towards the intersection) of both vehicles. Our model is represented as a Bayesian network in Figure 7 that shows the interaction between variables. The reward function of the model is constructed to take into account: comfort, velocity, time to collision, traffic rules and differences between intention and expectation. The system interacts with the environment by selecting an acceleration that maximize the current estimations of the sum of future expected rewards. Because of the stochastic aspect of the model and its solvers a safe intersection crossing cannot be guaranteed. Thus, a large number of simulations is required to validate the model in order to ensure the safety. The two problems

is that the scenario space is large because of the different regulations, initial speeds or different behaviours. Then, the parameter space for the model, especially its reward function, is as large and need to be correctly explored in order to find the functional range of the system.

B. Method Application

1) Dedicated simulator development: The decision-making system interacts with the simulation trough observations that can be made on the situation and selected actions that have to be realized in the simulated environment. Thus the fidelity, that is how closely the simulator can generate environmental data and model the system that are not under test, is important. In our scenario, the micro-traffic simulation (vehicle state and interactions between vehicles) is more important than the macro-simulation (simulation of traffic as a group of vehicles). As our system selects actions, it expects the other vehicle to change its behaviour. For the ego vehicle, the dynamic model of the vehicle does not need to have an high fidelity but as we want, in the future, to compare results obtained against field operational testing, the possibility of having high fidelity model is a plus. The decision could be of different forms (trajectory, goal points, control input), so the communication between the system under test and the simulation models must be adaptable. Figure 8 represents the different scenarios that have to be tested (yield, stop controlled, or priority). Thus the simulator must generate the appropriate behaviour for each of the corresponding situations. Real life scenarios could be also be imported to increase the validity of the reproduced situation. It would require the importation of maps and perception data from other sources. Scaner [33], an automotive grade simulator, has been chosen to test the decision-making systems. It has been mostly used for vehicle in the loop testing. However, most of the features previously described are available, at various levels of maturity. It has simple but interactive models for road intersection crossing and map generation. Scaner features a batch testing function, that we found too complex to interface with the SMC.

2) KPI definition: In order to evaluate the quality of the decision algorithm, we define some Key Performance Indicators regarding the crossing of a intersection. First, we define two areas in the intersection: a critical area, that correspond to the actual intersection where stopped vehicles would block all branches of the intersection and a non-critic area, that correspond to the entry of the intersection where cars usually stop before crossing the other road. We count the number and total duration of stops in each area, a smaller number indicates a better quality of the algorithm. We also measure the total time needed to cross the intersection, where again a smaller number indicates a better quality. We measure the acceleration to evaluate the comfort of the passenger, where again a smaller number indicates a better quality.

For all metrics m whose smaller value indicates a better performance, we check whether m is bounded by a bound b. The formula $G_{\leq t} m \leq b$, with t corresponding to the time needed to cross the intersection, states that m is always smaller than b. Stating that the acceleration must always be smaller than a bound might be a constraint too strong. We thus propose a relaxed version of this KPI where the acceleration is allowed to be above the bound for a short period of time (1s). This is stated by the formula $G_{\leq t} F_{\leq 1} acc \leq b$. The previous formula can be read as follows: at any point during the simulation, m will be smaller than b in less than 1s. In other words, it is not possible that $m > b$ for more than 1s. The value of the bound b is defined w.r.t. the metric considered.

Finally, to evaluate whether the intersection is crossed quickly enough, we set a maximum duration d for crossing the intersection and require that the intersection is crossed in less than d seconds, stated by $F_{\leq d} crossed$.

3) SMC application: In order to obtain results, we selected for each metric some adequate bounds and plot the probability that the KPI is met for each bound. The Figure 9 represents the probability that the acceleration/deceleration remains below a certain bound when crossing the intersection, both for the strict (i.e. the bound is never exceeded) and the relaxed version (the bound is never exceeded for more than 1s). We see that there is a probability 0 that the acceleration stays below an absolute value of $0.8 m.s^{-2}$, and that it is always below $2 m.s^{-2}$. It corresponds to an acceptable range for human comfort and shows that in every scenario the decision-making system took actions to adapt the behaviour.

Figures 10 and 11 present the probability of respectively having a bounded number of stops and having a bounded total stop duration. We see that there is a probability 0.9 that the

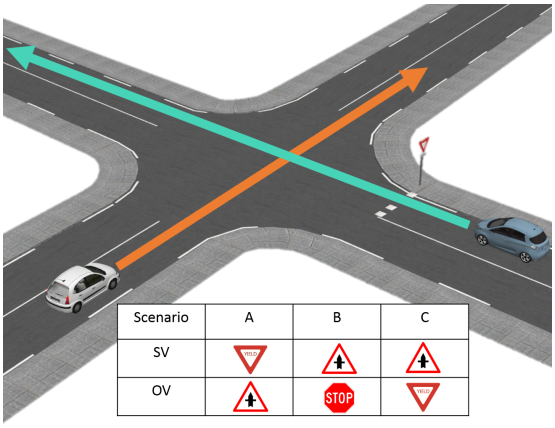


Figure 8. Simulated scenario for the decision-making. The ego vehicle (blue) is controlled by the decision-making system and have to interact with the other vehicle (white) with respect to the traffic rules

Table I
LIST OF VARIABLES EXTRACTED FROM THE SIMULATIONS.

Name	Description	Unit
t	Timestamp or time elapsed	s
nc_stops	Number of stops in the non-critical area	
c_stops	Number of stops in the critical area	
t_nc_stops	Duration of stops in non-critical area	s
t_c_stops	Duration of stops in critical area	s
acc	Acceleration	ms^{-2}
crossed	True if intersection is crossed	

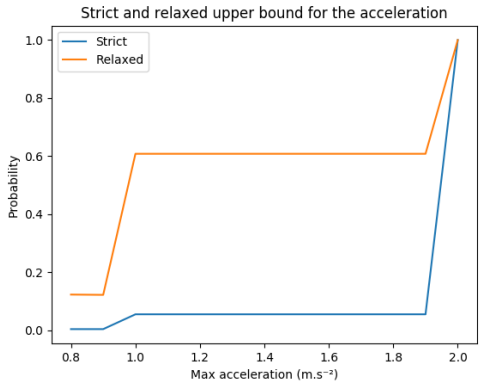


Figure 9. Probability that the absolute value of the acceleration remains bounded, for the strict and the relaxed version.

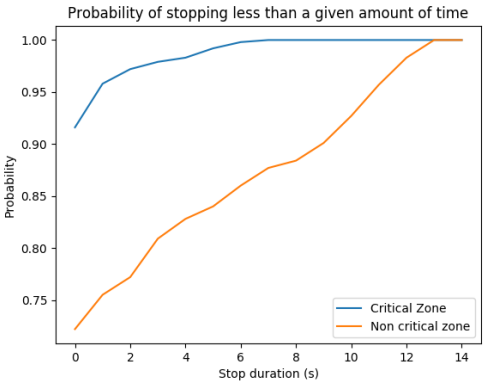


Figure 11. Probability of a stop duration below a given bound, for critical and non-critical zones.

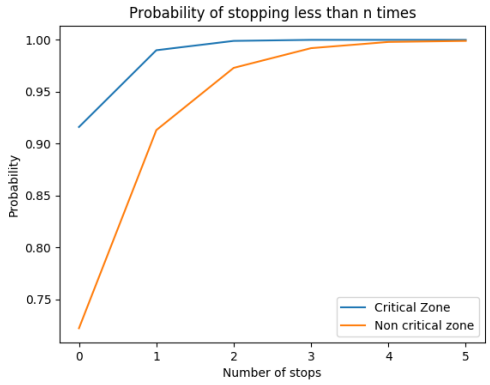


Figure 10. Probability of bounded occurences of stops, for critical and non-critical zones.

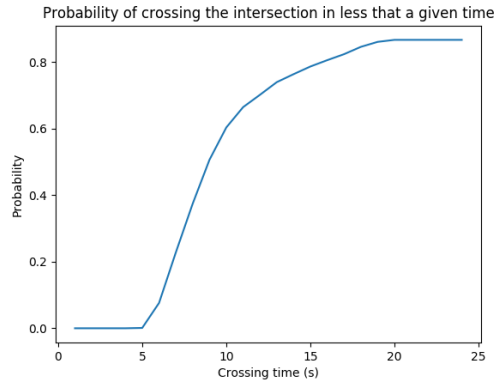


Figure 12. Probability of crossing the intersection in less than a given time.

car does not stop in the critical zone. With that measure it can be said that most likely the ego vehicle will comply with the traffic law. However for the 0.1 probability that the vehicle stop whitin the intersection, causes for the subject to come to a stop must be investigate in order to find if it correspond to an emergency manoeuvre or a failure of the system. This could be done by introducing finer KPIs that would take into account the temporality of the problem.

In Figure 12 we show the probability to cross the intersection in less that a given duration. All this new information tells people in charge of the validation what is the most likely behaviour of the decision-making system. It also helps people working on designing the decision-making to find area of improvement in their systems.

V. Conclusion

In this paper we presented and demonstrated a pipeline for the validation of different ACPS on two different automotive use-cases. The application of our approach based on Statistical Model Checking to the decision-making system provides useful information to the designer of the system and to the people in charge of the validation. This valuable information is formulated through probability for our system to stay in a certain range of KPIs.

Future works include the definition of meaningful grid-based metrics for stating more discriminating KPIs about the perception system. We also plan to compare results obtained in the simulated environment with tests on proving ground to ensure the validity of our approach. Also more KPIs for the decision and perception could be introduced to accurately pinpoint the cause of identified failures.

Acknowledgment

This work has been developed within the scope of ”Institut de Recherche Technologique” NanoElec, founded by the French program ”Investissements d’Avenir” ANR-10-AIRT-05.

This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL joint undertaking under grant agreement NO 692455. This joint undertaking receives support from the European Union’s Horizon 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

REFERENCES

[1] T. Bokc, M. Maurer, and G. Farber, "Validation of the vehicle in the loop (vil); a milestone for the simulation of driver assistance systems," in *2007 IEEE Intelligent Vehicles Symposium*, June 2007, pp. 612–617.

[2] T. Hwang, J. Roh, K. Park, J. Hwang, K. H. Lee, K. Lee, S. j. Lee, and Y. j. Kim, "Development of hils systems for active brake control systems," in *2006 SICE-ICASE International Joint Conference*, Oct 2006, pp. 4404–4408.

[3] J. Ibanez-Guzman, S. Lefevre, A. Mokkadem, and S. Rodhaim, "Vehicle to vehicle communications applied to road intersection safety, field results," in *13th International IEEE Conference on Intelligent Transportation Systems*, Sept 2010, pp. 192–197.

[4] Enable-S3, "Validation and testing of complex automated systems," https://www.enable-s3.eu/, 2016, last accessed 29/06/2018.

[5] M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Verification and validation of autonomous systems," in *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*, 2001, pp. 136–144.

[6] S. Ulbrich, D. Kappler, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann, "The opengrasp benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 1761–1767.

[7] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 719–726.

[8] Waymo, "Waymo s safety report: how we are building a safer driver," https://medium.com/waymo/waymos-safety-report-how-we-re-building-a-safer-driver-ce5f1b0d4c25, 2017, last accessed on 22/05/18.

[9] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[10] E. T. Jaynes, *Probability theory: the logic of science*. Cambridge university press, 2003.

[11] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet, "Approximate probabilistic model checking," in *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Implementations*, ser. Lecture Notes in Computer Science, B. Steffen and G. Levi, Eds. Germany: Springer Berlin Heidelberg, 2004, vol. 2937, pp. 73–84.

[12] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Proceedings of the 17th International Conference on Computer Aided Verification*, ser. Lecture Notes in Computer Science, K. Etessami and S. K. Rajamani, Eds. Germany: Springer Berlin Heidelberg, 2005, vol. 3576, pp. 266–280.

[13] K. Yi and J. Chung, "Nonlinear brake control for vehicle cw/ca systems," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 1, pp. 17–25, Mar 2001.

[14] A. Pnueli, "The temporal logic of programs," in *Proc. of the 18th Annual Symposium on Foundations of Computer Science*, ser. SFCS '77. Washington, DC, USA: IEEE Computer Society, 1977, pp. 46–57.

[15] P. Zuliani, A. Platzer, and E. M. Clarke, "Bayesian statistical model checking with application to stateflow/simulink verification," *Formal Methods in System Design*, vol. 43, no. 2, pp. 338–367, Oct 2013. [Online]. Available: https://doi.org/10.1007/s10703-013-0195-3

[16] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[17] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[18] P. Bessière, E. Mazer, J. Ahuactzin-Larios, and K. Mekhnacha, *Bayesian Programming*. CRC Press, Dec. 2013.

[19] M. Yguel, O. Aycard, and C. Laugier, "Efficient gpu-based construction of occupancy grids using several laser range-finders," in *International Journal of Vehicle Autonomous Systems*, vol. 6, 10 2006, pp. 105–110.

[20] L. Rummelhard, A. Négre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 2485–2490.

[21] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, vol. 19. IEEE, 1980, pp. 807–812.

[22] Z. Khan, T. Balch, and F. Dellaert, "An mcmc-based particle filter for tracking multiple interacting targets," in *Computer Vision-ECCV 2004*. Springer, 2004, pp. 279–290.

[23] R. Labayrade, C. Royere, and D. Aubert, "Experimental assessment of the rescue collision-mitigation system," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 1, pp. 89–102, Jan 2007.

[24] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, Jan 2009.

[25] L. Rummelhard, A. Nègre, M. Perrollaz, and C. Laugier, "Probabilistic grid-based collision risk prediction for driving application," in *International Synposium on Experimental Robotics*, Springer, Ed., Marrakech, Marocco, 2014.

[26] R. Grewe, M. Komar, A. Hohm, S. Lüke, and H. Winner, "Evaluation method and results for the accuracy of an automotive occupancy grid," *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pp. 19–24, 2012.

[27] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[28] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2164–2172. [Online]. Available: http://papers.nips.cc/paper/4031-monte-carlo-planning-in-large-pomdps.pdf

[29] W. Liu, S. W. Kim, S. Pendleton, and M. H. Ang, "Situation-aware decision making for autonomous driving on urban road using online pomdp," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1126–1133.

[30] M. Barbier, C. Laugier, O. Simonin, and J. Ibañez-Guzmán, "A pomdp-based intention-expectation decision-making and key performance indicators for road intersections crossing," 2018, submitted to IEEE Intelligent Vehicles Symposium (IV) 2018, under review.

[31] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Risk assessment at road intersections: Comparing intention and expectation," in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 165–171.

[32] M. Barbier, C. Laugier, O. Simonin, and J. Ibañez-Guzmán, "Classification of drivers manoeuvre for road intersection crossing with synthethic and real data," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 224–230.

[33] Oktal, "Scaner studio," http://www.oktal.fr/en/automotive/range-of-simulators/software, 2017, last accessed 05/02/2018.

Classifying Driver Behaviors for Autonomous Vehicle Navigation

Ernest Cheung[1], Aniket Bera[1], Emily Kubin[2], Kurt Gray[2], and Dinesh Manocha[1]

***Abstract*— We present a novel approach to automatically identify driver behaviors from vehicle trajectories and use them for safe navigation of autonomous vehicles. We propose a novel set of features that can be easily extracted from car trajectories. We derive a data-driven mapping between these features and six driver behaviors using an elaborate web-based user study. We also compute a summarized score indicating a level of awareness that is needed while driving next to other vehicles. We also incorporate our algorithm into a vehicle navigation simulation system and demonstrate its benefits in terms of safer real-time navigation, while driving next to aggressive or dangerous drivers.**

I. Introduction

Identifying dangerous drivers is crucial in developing safe autonomous driving algorithms and advanced driving assistant systems. The problem has been extensively studied in transportation and urban planning research [1]. However, prior work usually correlates driver' behaviors with their backgrounds (e.g., driver age, response to questionnaires, etc.). On the other hand, to develop autonomous vehicle systems, we need to understand the behavior of surrounding drivers using only the sensor data. As with to a human driver, an autonomous navigation algorithm that can predict other vehicle's driving behavior can navigate safely and efficiently avoid getting near dangerous drivers.

Prior work in transportation research [2], [1] often characterizes drivers using their levels of aggressiveness and carefulness. Several works in modeling pedestrian trajectories [3] and navigation [4] algorithms have applied psychological theory to capture human behavior. Current autonomous driving systems uses a range of different algorithms to process sensor data. Object detection and semantic understating methods are applied to obtain trajectory data [5]. Some work [6] uses end-to-end approaches to make navigation decisions from the sensor inputs (e.g. camera images, LIDAR data, etc.).

Main Results: We present a novel approach to automatically identifying driver behaviors from vehicle trajectories. We perform an extensive user study to learn the relationship and establish a mathematical mapping between extracted vehicular trajectories and the underlying driving behaviors: Trajectory to Driver Behavior Mapping (TDBM). TDBM enables a navigation algorithm to automatically classify the driving behavior of other vehicles. We also demonstrate simulated scenarios where navigating with our improved navigation scheme is safer.

Our approach takes into account different trajectory features. We use five different features, which can be easily extracted from vehicle trajectories and used to classify driving behaviors. We show that selecting a subset of these features is more favorable than selecting the currently used ones to produce a strong regression model that maps to driving behaviors.

As compared to prior algorithms, our algorithm offers the following benefits:

1. Driving Behavior Computation: We present a data-driven algorithm to compute TDBM. We conducted a comprehensive user survey to establish a mapping between five features and six different driving behaviors. We further conduct factor analysis on the six behaviors, which are derived from two commonly studied behaviors: aggressiveness and carefulness. The results show that there exists a latent variable that can summarize these driving behaviors and that can be used to measure the level of awareness that one should have when driving next to a vehicle. In the same study, we examine how much attention a human would pay to such a vehicle when it is driving in different relative locations.

2. Improved Realtime Navigation: We compute the features and identify the driving behaviors using TDBM. We enhance an existing Autonomous Driving Algorithm [7] to navigate according to the neighboring drivers' behavior. Our navigation algorithm identifies potentially dangerous drivers in realtime and chooses a path that avoids potentially dangerous drivers.

An overview of our approach is shown in Figure 1. The rest of the paper is organized as follows. We give a brief overview of prior work in Section II. We introduce the new trajectory features that are used to identify the driver behaviors in Section III. We present our data-driven mapping algorithm (TDBM) in Section IV and use it for autonomous car navigation in Section V.

II. Related Works

A. Studies on Driving Behaviors

There has been a wide range of work studying drivers' behaviors in Social Psychology and Transportation. Feng et al. [2] proposed five driver characteristics (age, gender, year of driving experience, personality via blood test, and education level) and four environmental factors (weather, traffic situation, quality of road infrastructure, and other cars' behavior), and mapped them to 3 levels of aggressiveness (driving safely, verbally abusing other drivers, and

[1]Authors from the Department of Computer Science, University of North Carolina at Chapel Hill, USA

[2]Authors from the Department of Psychology and Neuroscience, University of North Carolina at Chapel Hill, USA

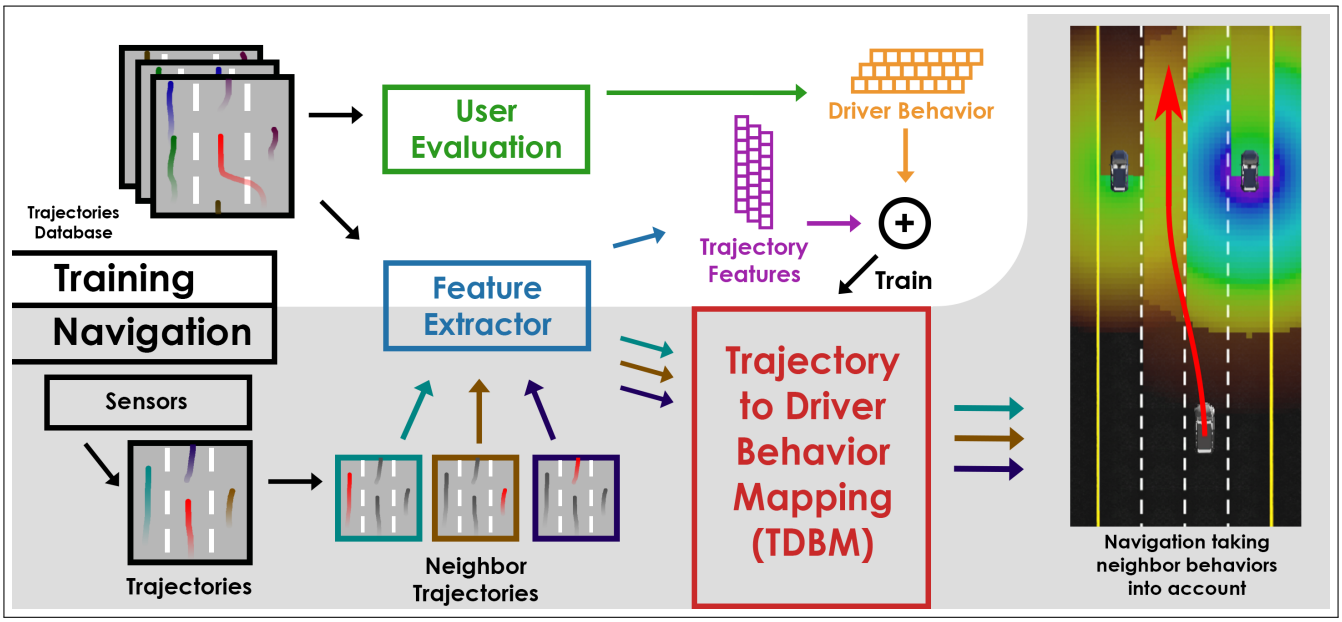


Fig. 1. Overview of our Algorithm: During the training of TDBM, we extract features from the trajectory database and conduct a user evaluation to find the mapping between them. During the navigation stage, we compute a set of trajectory and extract the features, then compute the driving behavior using TDBM. Finally, we plan for real-time navigation, taking into account these driver behaviors.

taking action against other drivers). Aljaafreh et al. [8] categorized driving behaviors into 4 classes: Below normal, Normal, Aggressive, and Very aggressive, in accordance to accelerometer data. Social Psychology studies [9], [10] have examined the aggressiveness according to the background of the driver, including age, gender, violation records, power of cars, occupation, etc. Meiring et al. [1] used several statistical reports to conclude that distracted behaviors and drunk behaviors are also serious threats to road safety. Many of the driver features used by these prior methods cannot be easily computed in new, unknown environments using current sensors. Our work uses trajectory data which can be extracted from sensor data in most autonomous driving systems and builds on the prior work [11], [12].

B. Trajectories Features

Murphey et al. [13] conducted an analysis on the aggressiveness of drivers and found that longitudinal (changing lanes) jerk is more related to aggressiveness than progressive (along the lane) jerk (i.e. rate of change in acceleration). Mohamad et al. [14] detected abnormal driving styles using speed, acceleration, and steering wheel movement, which indicate direction of vehicles. Qi et al. [15] studied driving styles with respect to speed and acceleration. Shi et al. [16] pointed out that deceleration is not very indicative of aggressiveness of drivers, but measurements of throttle opening, which is associated with acceleration, is more helpful in identifying aggressive drivers. Wang et al. [17] classified drivers into two categories, aggressive and normal, using speed and throttle opening captured by a simulator.

Instead of directly analyzing real-world data, many methods model driving behaviors as input parameters to generate driving simulations. Treiber et al. [18] proposed a lane following model, that controls the speed of the car using desired velocity, minimum spacing, desired time headway, acceleration, and maximum breaking deceleration. Kesting et al. [19] proposed a lane changing model, that makes lane changing decisions based on the speed advantage gained and the speed disadvantage imposed on the other vehicles, using a franticness and a politeness factor. Choudhury et al. [20] proposed a complex lane changing model, composed of desired speed, desired time gap, jam distance, maximum acceleration, desired deceleration, coolness factor, minimum acceptable gap, etc.

We combine a set of selected features proposed by previous works in terms of behavior mapping and simulation with two new trajectory features, lane following metric and relative speed metric. Then, we use variable selection to select a subset of features that can produce a good regression model.

C. Autonomous Car Navigation

There is substantial work on autonomous vehicle navigation [**?**], [21], [22], [23], [24], [25], [26], [27]. Ziegler et al. [28] presented a navigation approach that is capable of navigating through the historic Bertha Benz route in Germany. Numerous navigation approaches [29], [30], [31], [32] have been proposed in the DAPRA Urban Grand Challenge and the Grand Cooperative Driving Challenge. Recent work proposed by Best et al. [7], AutonoVi, presented an improved navigation algorithm that takes into account dynamic lane changes, steering and acceleration planning, and various other factors. Our approach is complimentary to these methods and can be combined with them.

D. Adaptation to Human Drivers' Behavior

Sadigh et al. [33] observed that an autonomous car's action could also affect neighboring human drivers' behavior, and studied how humans will react when the autonomous car performs certain actions [34]. Huang et al. [35] presented techniques for making autonomous car actions easily understandable to humans drivers. They also proposed an active learning approach [36] to model human driving behavior by showing examples of how a human driver will pick their preference out of a given set of trajectories. While this stream of work went further to take into account how humans would react to an autonomous car's action, it also emphasized the importance of a robot navigating according to other drivers' behavior.

III. Methodology

In this section, we present the two novel trajectory features that are used to identify driver behaviors. We also compare their performance with other features and give an overview of driver behavior metrics used in our navigation algorithm.

A. Features

The goal of our work is to extract a set of trajectory features that can be mapped properly to driving behaviors. We assume that the trajectories have been extracted from the sensor data. Many of the previous works deal with different driver characteristics: driver background, accelerometer use, throttle opening, etc., which may not be available for an autonomous vehicle in new and uncertain environments. Moreover, in the simulation models described in Section II-B, a lot of features cannot be measured from trajectories with insufficient lane-changing samples: comfortable breaking deceleration, desired time headway, etc. Therefore, we derive some variants of features that can be easily extracted from the trajectories and summarize them in Table I. These features are further shortlisted with the results from a user study described in the next section.

Symbol	Notation	Description
f_0	v_{front}	Average relative speed to the car in front
f_1	v_{back}	Average relative speed to the car in the back
f_2	v_{left}	Average relative speed to cars in the left lane
f_3	v_{right}	Average relative speed to cars in the right lane
f_4	v_{nei}	Relative speed to neighbors
f_5	v_{avg}	Average velocity
f_6	s_{front}	Distance with front car
f_7	j_l	Longitudinal jerk
f_8	j_p	Progressive jerk
f_9	s_{center}	Lane following metric

TABLE I

We considered ten candidate features $f_0, .., f_9$ for selection. Features highlighted in green are selected for mapping to behavior-metrics only, and those in blue are selected for mapping to both behavior-metrics and attention metrics.

1) Acceleration: As pointed out in several prior works [13], [14], [16], [17], acceleration is often correlated with driver aggressiveness. While previous studies [13] concluded that longitudinal jerk can reflect aggressiveness better than progressive jerk, our goal is to use features that also correlate with all the driving styles, instead of just aggressiveness. Therefore, we include both longitudinal jerk j_l and progressive jerk j_p in our computations.

2) Lane following: Previous work [37] proposed a metric measureing the extent of lane following that depends on the mean and standard deviation of lane drifting and lane weaving. We propose a feature that also depends on lane drifting, but distinguishes between drivers who keep drifting left and right within a lane and those who are driving straight but not along the center of the lane. Moreover, we compensate for the extent of lane drifting while performing lane changing to avoid capturing normal lane changing behaviors into this metric.

Given y_l, which is the center longitudinal position of the lane that the targeted car is in, and $y(t)$, which is the longitudinal position of the car at time t, we detect a lane changing event when the car has departed from one lane to the another and remained in the new lane for at least k seconds.

With a set of changing lane events happened at time t_i, $C = \{t_1, t_2, ..., t_n\}$, the lane drift metric $s_C(t)$ is measured as below:

$$s_C(t) = \begin{cases} 0, & \text{if } \exists t \in C \text{ s.t. } t \in [t-k, t+k], \\ y(t) - y_l, & \text{otherwise.} \end{cases} \quad (1)$$

We use a term that measures the previous τ seconds of rate of change in drifting to differentiate lane drifts from those drivers who are driving straight but off the center of the lane. Our overall lane following metric is illustrated in Figure 2 and defined as:

$$s_{center} = \int |s_C(t)| \left[\mu + \int_{t-\tau}^{t} |s'_{\emptyset}(t)| dt \right] dt, \quad (2)$$

where μ is a parameter that distinguish drivers who are driving off the center of the lane and those who are along.

3) Relative Speed: Relative speed has been used to evaluate the aggressiveness of drivers [15]. However, directly measuring the relative speed using v_{front}, v_{back}, v_{left} and v_{right} has many issues. First, such a feature sometimes does not exist as there may be no car next to the target car. Second, these features might not be directly related to the driving behavior of the car. While driving substantially faster than other cars would be perceived as aggression, driving slower might not necessarily imply that the driver is non-aggressive. Third, computing such an average velocity requires knowledge about the trajectories and range of speeds of the neighboring vehicles. Given these considerations, we design the following metric to capture the relationship between the driving behavior and the relative speed with respect to neighboring cars:

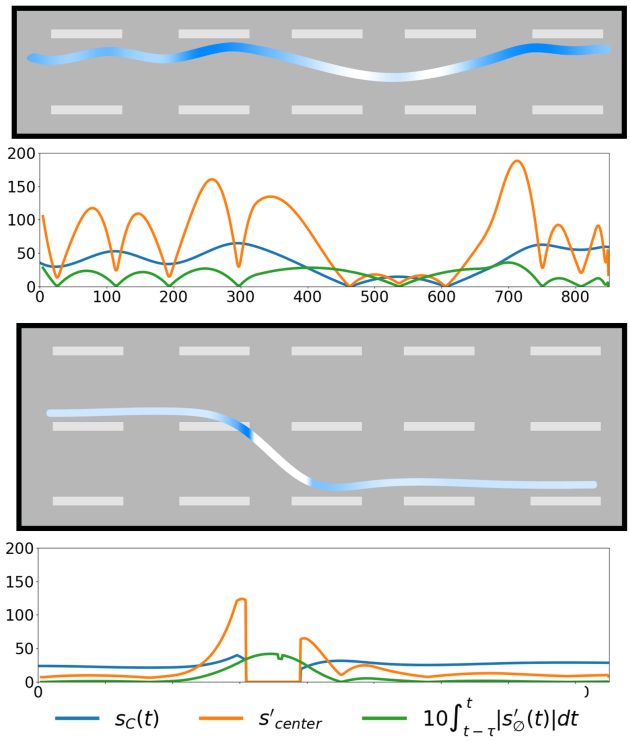


Fig. 2. Illustration of the lane drift metric ($|s_c(t)|$), and the lane following metric (s_{center}). The lane following metric for the trajectories above is the sum of the area under the plot of s'_{center}. This two example shows that our lane following metric (s_{center}) captures the 'drifting behavior' in the top example, but not the 'driving straight off the center' and 'lane changing' shown in the bottom example.

$$v_{nei} = \int \sum_{n \in N} \max(0, \frac{v(t) - v_n(t)}{dist(x(t), x_n(t))}) dt, \qquad (3)$$

where N denotes the set containing all neighboring cars within a large range (e.g., a one-mile radius). $x(t)$, $v(t)$, $x_n(t)$, $v_n(t)$ denote the position and the speed of the targeting car, and the position and the speed of the neighbor n, respectively.

B. Driving Behavior Metrics

As discussed in Section II-A, aggressiveness [2], [8], [38] and carefulness [1], [39], [40] are two metrics that have been used to identify road safety threats. Typically, social psychologists add related items into studies to leverage robustness and the observed effects. Therefore, we would like to evaluate four more driving behaviors: Reckless, Threatening, Cautious, and Timid. They are listed in Table II.

C. Attention Metrics

Observing different maneuvers of other drivers on the road can result in paying more attention to those drivers. However, the relative position of such drivers (with respect to the targeted vehicle) would affect the level of attention that one is paying to them. For instance, one would pay more attention to a vehicle in the front making frequent stops, as opposed to a following vehicle. We would like to understand how much attention a driver will pay to the targeted car when the user assumes that he or she is driving in different relative positions than the target. We study four different relative positions: preceding, following, adjacent to and far away from the targeted vehicle, also listed in Table II. These positions affect the level of attention one would pay when driving in that relative position.

Symbol	Description	Symbol	Level of Attention when
b_0	Aggressive	b_6	following the target
b_1	Reckless	b_7	preceding the target
b_2	Threatening	b_8	driving next to the target
b_3	Careful	b_9	far from the target
b_4	Cautious		
b_5	Timid		

TABLE II
SIX DRIVING BEHAVIOR METRICS (b_0, b_1, ...,b_5) AND FOUR ATTENTION METRICS (b_6, b_7, b_8, b_9) USED IN TDBM

IV. DATA-DRIVEN MAPPING

We designed a user study, involving 100 participants to identify driver behaviors from videos rendered from the Interstate 80 Freeway Dataset [41]. The video dataset consists of 45 minutes of vehicle trajectories, captured in a 1650 feet section on I-80 in California, US. The videos were first annotated automatically using a proprietary code developed in the NGSIM program, and then manually checked and corrected. The raw videos provided in the dataset are low in quality and divided into seven different segments with different camera angles. Therefore, we have rendered the videos using a game engine, Unreal Engine, to provide a stable and consistent view for the users in the survey. The virtual cameras have a fixed transform to the targeted car, which is highlighted in red, and will follow it throughout the video.

Figure 3 shows snapshots of the videos used in the user study. The participants were asked to rate the six behaviors we described in Section III-B on a 7-point scale: {Strongly disagree, Disagree, Somewhat disagree, Neither agree or disagree, Somewhat agree, Agree, Strongly agree}. This was followed by another question on how much attention they would be paying if they were in different positions relative to the targeted car, as described in Section III-C, on a 5-point scale, where -2 indicates not at all, 0 indicates a moderate amount and 2 indicates a lot.

A. Data Pre-Processing

We perform data augmentation to make sure that the dataset has a sufficiently wide spectrum of driving behaviors corresponding to lane changes, fast moving cars, passing cars, etc. In addition, the features described in Table I are measured using different units. To improve numerical stability during the regression analysis, we scale the data linearly using the 5th and the 95th percentile samples.

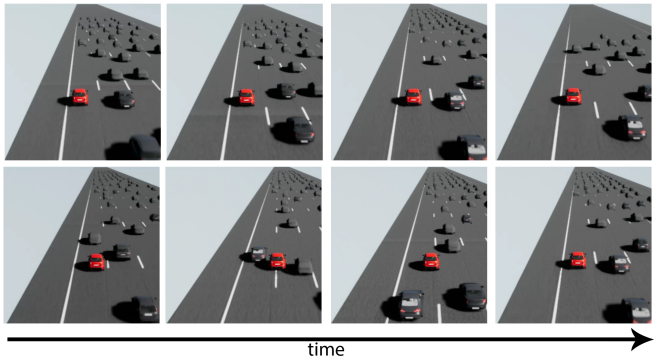


Fig. 3. Two example videos used in the user study. Participants are asked to rate the six driving behavior metrics and four attention metrics of the target car colored in red.

	v_{front}	v_{left}	v_{right}	v_{back}	v_{nei}	v_{avg}	s_{front}	j_l	j_p	s_{center}
Aggressive	-2.7	-2.08	-2.12	-1.78	-2.2	-0.65	-0.62	-1.45	-1.72	-2.05
Reckless	-2.7	-1.62	-3.55	-2.22	-1.35	-1.85	-1.38	-1.7	-3.0	-0.38
Threatening	-1.78	-1.92	-1.92	-2.92	-1.5	-1.72	-2.1	-2.1	-2.4	-0.45
Careful	-1.82	-2.52	-1.92	-2.92	-1.35	-2.58	-2.02	-2.2	-2.48	-0.45
Cautious	-2.05	-2.8	-2.2	-3.05	-1.48	-2.55	-2.0	-2.17	-2.95	-0.52
Timid	-2.17	-1.95	-2.4	-2.55	-1.52	-2.02	-2.0	-2.05	-2.85	-0.52
$Attention_{back}$	-3.35	-1.62	-2.8	-1.88	-1.58	-1.72	-1.88	-2.2	-2.9	-0.52
$Attention_{front}$	-2.62	-3.32	-2.75	-4.5	-1.78	-2.12	-2.33	-2.1	-2.48	-0.75
$Attention_{adj}$	-2.02	-2.12	-3.52	-2.35	-1.78	-1.32	-2.15	-2.5	-2.22	-1.08
$Attention_{far}$	-1.98	-2.4	-3.72	-2.45	-1.75	-1.78	-2.2	-2.48	-2.65	-0.9
Behavior Average	-2.2	-2.15	-2.35	-2.58	-1.57	-1.9	-1.69	-1.95	-2.57	-0.73
Attention Average	-2.49	-2.37	-3.2	-2.79	-1.72	-1.74	-2.14	-2.32	-2.56	-0.81
Overall Average	-2.32	-2.24	-2.69	-2.66	-1.63	-1.83	-1.87	-2.1	-2.56	-0.76

Fig. 4. The converging value (in the power of 10) of $\beta_{i,j}$ which maps a feature f_j to a behavior/attention metric b_i while performing Lasso analysis. A larger converging value indicates a higher likelihood that the feature is favourable in regression analysis, and therefore we select that value for TDBM.

B. Feature Selection

In Section III-A and Table I, we cover a wide range of features used in previous studies that can be extracted from trajectories, along with two new metrics that attempt to summarize some of these features to avoid strong correlation between independent variables during regression analysis. In this section, we apply feature selection techniques to find out which features are most relevant to the driving behaviors.

We perform least absolute shrinkage and selection operator (Lasso) analysis on six driving behaviors b_0, b_1, ..., b_5 and four attention metrics, b_6, b_7, b_8, b_9, from the user responses. The objective function for Lasso analysis conducted on b_i is:

$$\min_{\beta_i',\beta_i} \left[\frac{1}{N} \sum_{j=1}^{N} (b_i - \beta i' - f_j^T \beta_{i,j}) \right], \text{ subject to } \sum_{j=1}^{F} |\beta_{i,j}| \leq \alpha_i, \tag{4}$$

where N is the number of survey responses and F is the number of features.

Lasso analysis performs regularization and feature selection by eliminating weak subsets of features. The parameter α_i determines the level of regularization that Lasso analysis imposes on the features. As we increase α_i, features f_j will be eliminated in a different order. Unlike regular regression analysis on a single dependent variable, our goal is to select two sets of features: one that can produce a strong regression model for all six driving behavior metrics, and one for all four attention metrics. We sample different values of α_i for all responses b_i, and record the values of α_i at which the component $\beta_{i,j}$ (which mapping feature f_j to response b_i) converges to 0. The results are shown in Figure 4, where converging values of $\beta_{i,j}$ are presented in the power of 10.

The directly computed relative speeds of the cars surrounding the targeted car are least favorable for selection for both regressions for behavior-metrics and attention-metrics. However, our relative speed metric proposed to capture the correlation between surrounding cars and the targeted car, v_{nei} (Equation 3), is more favorable in terms of being selected. Moreover, our lane following metric, s_{center} (Equation 2), tends to be the last one eliminated as a feature in the variable selection stage.

Our goal is to find two $\alpha_{behavior}$ and $\alpha_{attention}$ that shortlist a subset of features for behavior-metric and attention-metric respectively. Note that $\alpha_{behavior} = \alpha_i, \forall i \in [0, 5]$, and $\alpha_{attention} = \alpha_i, \forall i \in [6, 9]$ for α_i defined in Equation 4. In terms of behavior, we can either pick $\{s_{center}, v_{nei}, s_{front}\}$ or $\{s_{center}, v_{nei}, s_{front}, v_{avg}, j_l\}$. Given that the mapping component between v_{avg} and j_l has high converging values, they can produce a stronger regression model for aggression, and that aggressiveness is one of the common behaviors as studied in prior literature discussed in Section II-A. We therefore select the latter set of features for behavior mapping. For mapping features with attention regions metrics, we select $\{s_{center}, v_{nei}, v_{avg}\}$.

C. Feature-Behavior Mapping

Using $\{s_{center}, v_{nei}, s_{front}, v_{avg}, j_l\}$ and $\{s_{center}, v_{nei}, v_{avg}\}$ as the features, we perform linear regression to obtain the mapping between these selected features and the drivers' behavior. We normalize the data as described in Section IV-A to increase the numerical stability of the regression process. The results we obtained are below. For $B_{behavior} = [b_0, b_1, ..., b_5]^T$, we obtain

$$B_{behavior} = \begin{pmatrix} 1.63 & 4.04 & -0.46 & -0.82 & 0.88 & -2.58 \\ 1.58 & 3.08 & -0.45 & 0.02 & -0.10 & -1.67 \\ 1.35 & 4.08 & -0.58 & -0.43 & -0.28 & -1.99 \\ -1.51 & -3.17 & 1.06 & 0.51 & -0.51 & 1.39 \\ -2.47 & -2.60 & 1.43 & 0.98 & -0.82 & 1.27 \\ -3.59 & -2.19 & 1.75 & 1.73 & -0.30 & 0.61 \end{pmatrix} \begin{pmatrix} s_{center} \\ v_{nei} \\ s_{front} \\ v_{avg} \\ j_l \\ 1 \end{pmatrix} \tag{5}$$

Moreover, for $B_{attention} = [b_6, b_7, b_8, b_9]^T$,

$$B_{attention} = \begin{pmatrix} B_{back} \\ B_{front} \\ B_{adj} \\ B_{far} \end{pmatrix} = \begin{pmatrix} 0.54 & 1.60 & 0.11 & -0.8 \\ -0.73 & 1.66 & 0.63 & -0.07 \\ -0.14 & 1.73 & 0.25 & 0.15 \\ 0.25 & 1.47 & 0.17 & -1.43 \end{pmatrix} \begin{pmatrix} s_{center} \\ v_{nei} \\ v_{avg} \\ 1 \end{pmatrix} \tag{6}$$

We further apply leave-one-out cross-validation to the set of samples S: enumerate through all samples $s_i \in S$ and leave s_i as a validation sample, and use the remaining samples $S - s_i$ to produce regression models $M_{i,j}$ for each behavior $b_{i,j}$. Using $M_{i,j}$, we predict the behaviors $b_{i,j}$ of s_i. The mean prediction errors of $b_{i,j}$ using $M_{i,j}$ are listed in the table below. The mean prediction error in the cross-validation is less than 1 in a 7-point scale for all behaviors and attention metrics predicted, showing that our mappings described in Equation 5 and 6 are not over-fitted.

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0.75	0.94	0.78	0.7	0.6	0.89	0.2	0.49	0.38	0.23

TABLE III

MEAN ERROR IN A 7-POINT SCALE WHEN APPLYING CROSS VALIDATION OF LINEAR REGRESSION TO MAP FEATURE TO BEHAVIOR AND ATTENTION METRICS SHOWING OUR MAPPING IS NOT OVER-FITTED.

D. Factor Analysis

Previous studies on mapping walking behavior adjectives with features used to simulate crowds [3], have applied factor analysis to find smaller numbers of primary factors that can represent the personalities or behaviors. We can apply Principal Component Analysis (PCA) to the survey response. The percentages of variance of the principal components are 73.42%, 11.97%, 7.78%, 2.96%, 2.30% and 1.58%. The results indicate that the Principal Component 1, which has variance of 73.43%, can model most of the driving behaviors.

We represent each entry of the user study response with the highest rated behavior and transform these entries into the space of the Principal Components as shown in Figure 5. If the user did not fully agree to any behavior for a video (i.e. responses to all questions are below 'Somewhat agree'), we consider that there to be no representative behavior from this entry (i.e. undefined). Also, if a response indicates more than one behavior as the strongest, then we label those behaviors as undefined if those adjectives contradict each other (i.e. one from negative adjectives {Aggressive, Reckless, Threatening} and one from positive adjectives {Careful, Cautious, Timid}). As observed in Figure 5, the distribution of the data on Principal Component 1, the three negative behavior adjectives we used in the user study, represented in warmer colors, are distributed on the negative side, while the three positive behavior adjectives are distributed on the positive side. Furthermore, the entries that suggest the users' responses were 'Strongly agree', represented by solid color plots in Figure 5, have significantly higher magnitudes in terms of value along Principal Component 1. However, for Principal Components 2 and 3, such a relationship is not observed.

Our studies show that there could be one latent variable that is negatively correlated with aggressiveness and positively correlated with carefulness. We further verify these results by analyzing the correlation of the Principal Components with the amount of awareness that the users indicated they would pay to the targeted car. We take the average of the level of attention, $\frac{b_6+b_7+b_8+b_9}{4}$, recorded for each response and plot these averages as the color on the PCA results in Figure 6. Similar results have been observed from this user evaluation, where the drivers worth more attention have a lower value of Principal Component 1, and those who worth less attention tend to have a higher value. Moreover, there is no clear evidences pointing to correlation between the level of awareness the user rated and Principal Component 2 or 3.

Therefore, we consider the Principal Component 1 as a

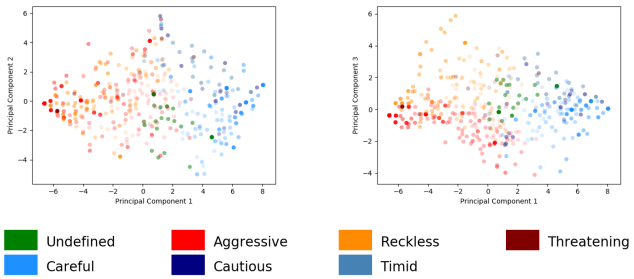


Fig. 5. Principal Component Analysis results for {Principal Components 1(PC1), PC2} (left) and {PC1, PC3} (right). The color of the data point indicates the highest rated driving behavior adjective as shown in the legends, and the alpha value indicates the rating of this behavior (solid for 'Strongly agree', and half-transparent for 'Somewhat agree'). If a user did not agree to any of the behaviors or indicated multiple contradicting behaviors, the data point is marked as undefined in green.

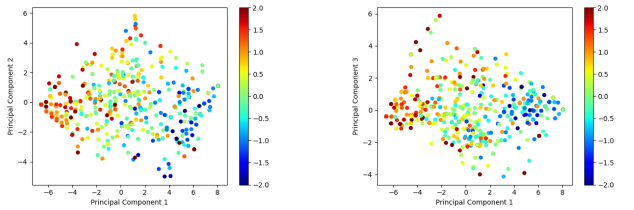


Fig. 6. Principal Component Analysis results for {PC1, PC2} (left) and {PC1, PC3} (right). The color of the data point indicates the average amount of awareness the user rated on a 5 point scale (-2 for not paying any attention at all, and 2 for paying a lot of attention).

safety score reflecting the amount of attention awareness that a driver or an autonomous navigation system should take into account. TDBM is therefore computed as below:

$$S_{TDBM} = \begin{pmatrix} -4.78 & -7.89 & 2.24 & 1.69 & -0.83 & 4.69 \end{pmatrix} \begin{pmatrix} s_{center} \\ v_{nei} \\ s_{front} \\ v_{avg} \\ jl \\ 1 \end{pmatrix} \quad (7)$$

V. NAVIGATION

In this section, we highlight the benefits of identifying driver behaviors and how these ensure safe navigation. We extend an autonomous car navigation algorithm, AutonoVi [7], and show improvements in its performance by using our driver behavior identification algorithm and TDBM. AutonoVi is based on a data-driven vehicle dynamics model and optimization-based maneuver planning, which generates a set of favorable trajectories from among a set of possible candidates, and performs selection among this set of trajectories using optimization. It can handle dynamic lane-changes and different traffic conditions.

The approach used in AutonoVi is summarized below: The algorithm takes a graph of roads from a GIS database, and applies A* algorithm to compute the shortest global route plan. The route plan consists of a sequence of actions that is composed of {Drive Straight, Turn Left, Turn Right, Merge Left, and Merge Right}. The plan is translated to a static

guiding path that consists of a set of way-points, that exhibits C^1 continuity, and that takes Traffic Rules into account (e.g., making a stop at an intersection). AutonoVi then samples the steering angle and velocity in a desirable range of values to compute a set of candidate trajectories, and eliminates the trajectories that lead to possible collisions based on Control Obstacles [42].

Among the set of collision-free trajectories, AutonoVi selects the best trajectory by optimizing a heuristic that penalizes trajectories that lead to: i) deviation from global route; ii) sharp turns, braking and acceleration; iii) unnecessary lane changes; and iv) getting too close to other vehicles and objects (even without a collision).

To avoid getting too close to other neighboring entities, AutonoVi proposed a proximity cost function to differentiate entities only by its class. That is, it considers all vehicles as the same and applies the same penalization factor, $F_{vehicle}$, to them. Further, it applies a higher factor : F_{ped} and F_{cyc} to pedestrians and cyclist respectively. The original proximity cost used in AutonoVi is:

$$c_{prox} = \sum_{n=1}^{N} F_{vehicle}\, e^{-d(n)} \tag{8}$$

This cost function has two issues: i) it cannot distinguish dangerous drivers to avoid driving too close to them, and ii) it diminishes too rapidly due to its use of an exponential function. We propose a novel proximity cost that can solve these problems:

$$c'_{prox} = \sum_{n=1}^{N} c(n) \tag{9}$$

$$c(n) = \begin{cases} 0 & \text{if } d \in [d_{t2}, \inf), \\ S_{TDBM} B_{far} \frac{d_{t2} - d(n)}{d_{t2}} & \text{if } d \in (d_t, d_{t2}], \\ S_{TDBM} \left[\frac{(d_t - d(n))(B_r - B_{far})}{d_t} + B_{far} \right] & \text{if } d \in (0, d_t]. \end{cases} \tag{10}$$

where $d(n)$ is the distance between the car navigating with our approach and the neighbor n, d_t is a threshold distance beyond which neighbors are considered as far away, and d_{t2} is a threshold distance beyond which neighbors would no longer have impact on our navigation. S_{TDBM} is derived from Equation 7, B_{far} and B_r are the attention metrics computed using the features extracted from the features using the mapping in equation 6, for r = {back, front, adj} if the neighboring car is following, preceding, and next to the navigating car, respectively.

Using this new cost function, we can avoid drivers that are potentially riskier, and select a safer navigation path. Examples of scenarios are illustrated in Figure 7 and the attached video.

VI. Conclusion and Future Works

We present a novel data-driven approach to enable safer real-time navigation by identifying human drivers who are potentially hazardous. Our studies and findings are based on a data-driven mapping computation (TDBM). We conclude that although humans use different adjectives when describing driving behavior, there is an underlying latent variable, S_{TDBM} (Equation 7), that reflects the level of attention humans pay to other vehicles' driving behavior. Moreover, we can estimate this variable by a set of novel trajectory features and other existing features.

Current trajectory data tends to be limited due to human labeling or the fact that extra efforts may be needed to extract such annotated data from raw images. With advancement in object detection and other work in computer vision, one can expect more trajectory data would be made available to the autonomous driving research community. Given more variety of data (e.g., in urban environments or different cultures), we would like to apply our approach to analyzing and developing different navigation strategies that adapt to these new situations and local driving styles.

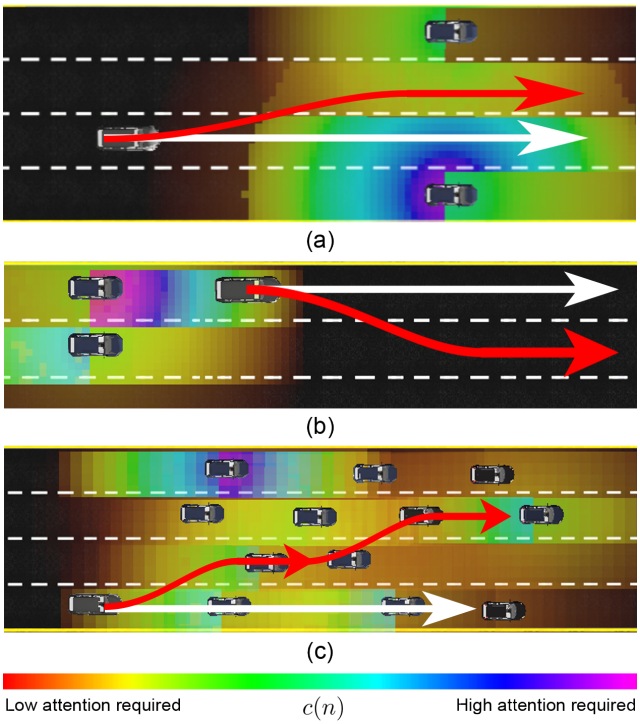


Fig. 7. Examples of our approach making better navigation decision than AutonoVi. The red route is the one selected by our approach while white is the one selected by AutonoVi. The cost map $c(n)$ is also shown for each neighbor car n indicating the amount of attention needed. In (a), our algorithm chooses to switch lane and keep a distance from the car require more attention. In (b), a car requiring high level of attention tailgates the car running our approach, and we switch to a slower lane to give way. In (c), a heavy traffic ahead causing all four lanes move at a similarly low speed, and our algorithm chooses to the follow the car with the lowest attention required.

References

[1] G. A. M. Meiring and H. C. Myburgh, "A review of intelligent driving style analysis systems and related artificial intelligence algorithms," *Sensors*, vol. 15, no. 12, pp. 30 653–30 682, 2015.

[2] Z.-X. Feng, J. Liu, Y.-Y. Li, and W.-H. Zhang, "Selected model and sensitivity analysis of aggressive driving behavior," *Zhongguo Gonglu Xuebao(China Journal of Highway and Transport)*, vol. 25, no. 2, pp. 106–112, 2012.

[3] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, 2011, pp. 43–52.

[4] A. Bera, T. Randhavane, and D. Manocha, "Aggressive, tense, or shy? identifying personality traits from crowd videos," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[7] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha, "Autonovi: Autonomous vehicle planning with dynamic maneuvers and traffic constraints," *arXiv:1703.08561*, 2017.

[8] A. Aljaafreh, N. Alshabatat, and M. S. N. Al-Din, "Driving style recognition using fuzzy logic," in *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pp. 460–463.

[9] B. Krahé and I. Fenske, "Predicting aggressive driving behavior: The role of macho personality, age, and power of car," *Aggressive Behavior*, vol. 28, no. 1, pp. 21–29, 2002.

[10] K. H. Beck, B. Ali, and S. B. Daughters, "Distress tolerance as a predictor of risky and aggressive driving," *Traffic injury prevention*, vol. 15, no. 4, pp. 349–354, 2014.

[11] E. Cheung, A. Bera, and D. Manocha, "Efficient and safe vehicle navigation based on driver behavior classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[12] E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha, "Identifying driver behaviors using trajectory features for vehicle navigation," in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*, 2018.

[13] Y. L. Murphey, R. Milton, and L. Kiliaris, "Driver's style classification using jerk analysis," in *Computational Intelligence in Vehicles and Vehicular Systems, 2009. CIVVS'09. IEEE Workshop on*, pp. 23–28.

[14] I. Mohamad, M. A. M. Ali, and M. Ismail, "Abnormal driving detection using real time global positioning system data," in *Space Science and Communication (IconSpace), 2011 IEEE International Conference on*, pp. 1–6.

[15] G. Qi, Y. Du, J. Wu, and M. Xu, "Leveraging longitudinal driving behaviour data with data mining techniques for driving style analysis," *IET intelligent transport systems*, vol. 9, no. 8, pp. 792–801, 2015.

[16] B. Shi, L. Xu, J. Hu, Y. Tang, H. Jiang, W. Meng, and H. Liu, "Evaluating driving styles by normalizing driving behavior based on personalized driver modeling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1502–1508, 2015.

[17] W. Wang, J. Xi, A. Chong, and L. Li, "Driving style classification using a semisupervised support vector machine," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 650–660, 2017.

[18] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[19] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1999, pp. 86–94, 2007.

[20] C. F. Choudhury, "Modeling lane-changing behavior in presence of exclusive lanes," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[21] A. Bera, T. Randhavane, E. Kubin, A. Wang, K. Gray, and D. Manocha, "The socially invisible robot: Navigation in the social world using robot entitativity," in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*, 2018.

[22] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[23] M. Saifuzzaman and Z. Zheng, "Incorporating human-factors in car-following models: a review of recent developments and research needs," *Transportation research part C: emerging technologies*, vol. 48, pp. 379–403, 2014.

[24] A. Bera, S. Kim, T. Randhavane, S. Pratapa, and D. Manocha, "Glmp-realtime pedestrian path prediction using global and local movement patterns," *ICRA*, 2016.

[25] S. Kolski, D. Ferguson, M. Bellino, and R. Siegwart, "Autonomous driving in structured and unstructured environments," in *Intelligent Vehicles Symposium*. IEEE, 2006, pp. 558–563.

[26] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *American Control Conference ACC'07*. IEEE, 2007, pp. 2296–2301.

[27] A. Bera, S. Kim, and D. Manocha, "Modeling trajectory-level behaviors using time varying pedestrian movement dynamics," *Collective Dynamics*, vol. 3, pp. 1–23, 2018.

[28] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[29] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[30] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler, "Team annieway's entry to the 2011 grand cooperative driving challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1008–1017, 2012.

[31] R. Kianfar, B. Augusto, A. Ebadighajari, U. Hakeem, J. Nilsson, A. Raza, R. S. Tabar, N. V. Irukulapati, C. Englund, P. Falcone *et al.*, "Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge," *IEEE transactions on intelligent transportation systems*, vol. 13, no. 3, pp. 994–1007, 2012.

[32] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, 2016.

[33] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, 2016.

[34] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 66–73.

[35] S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan, "Enabling robots to communicate their objectives," *arXiv:1702.03465*, 2017.

[36] A. D. D. Dorsa Sadigh, S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems (RSS)*, 2017.

[37] L. M. Bergasa, D. Almería, J. Almazán, J. J. Yebes, and R. Arroyo, "Drivesafe: An app for alerting inattentive drivers and scoring driving behaviors," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 240–245.

[38] P. B. Harris, J. M. Houston, J. A. Vazquez, J. A. Smither, A. Harms, J. A. Dahlke, and D. A. Sachau, "The prosocial and aggressive driving inventory (padi): A self-report measure of safe and unsafe driving behaviors," *Accident Analysis & Prevention*, vol. 72, no. Supplement C, pp. 1 – 8, 2014.

[39] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia, "Data-driven probabilistic modeling and verification of human driver behavior," 2014.

[40] M. Lan, M. Rofouei, S. Soatto, and M. Sarrafzadeh, "Smartldws: A robust and scalable lane departure warning system for the smartphones," in *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pp. 1–6.

[41] J. Halkia and J. Colyar, "Interstate 80 freeway dataset," *Federal Highway Administration, U.S. Department of Transportation*, 2006.

[42] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Perception

- **Keynote speaker: Alberto Broggi (Ambarella/USA, VisLab/Italy)**
  Title: SoC for ultra HD mono and stereo-vision processing

- **Title: Intelligent feature selection method for accurate laser-based mapping and localisation in self-driving**
  Authors: N. Hernandez, I. G. Daza, C. Salinas, I. Parra, J. Alonso, D. Fernandez-Llorca, M.A. Sotelo

- **Title: LiDAR based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape**
  Authors: E. Héry, P. Xu and P. Bonnifait

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Keynote speaker: **: Alberto Broggi**
(Ambarella/USA, VisLab/Italy)

SoC for ultra HD mono and stereo-vision processing

Abstract: Ambarella has designed a sensing suite for autonomous vehicles that relies on input from stereovision and monocular cameras. The cameras, each based on the new Ambarella CV1 SoC (System on Chip), provide a perception range of over 150 meters for stereo obstacle detection and over 180 meters for monocular classification to support autonomous driving and deliver a viable, high-performance alternative to lidar technology.
Ambarella's solution not only recognizes visual landmarks, but also detects obstacles without training and runs commonly used CNNs for classification. Additional features include automatic stereo calibration, terrain modeling, and vision-based localization.
The presentation will emphasize the underlying architecture of Ambarella's CV1-based solution for vision-based autonomous vehicle driving systems and provide insight into the main technological breakthrough that made it possible.

Biography: Prof. Alberto Broggi received the Dr. Ing. (Master) degree in Electronic Engineering and the Ph.D. degree in Information Technology both from the Universita` di Parma, Italy. He is now Full Professor at the Universita` di Parma and the President and CEO of VisLab, a spinoff company focused on visual perceptioin for intelligent vehicles. As a pioneer in the use of machine vision for automotive applications and on driverless cars, he authored of more than 150 publications on international scientific journals, book chapters, refereed conference proceedings. He served as Editor-in-Chief of the IEEE Transactions on Intelligent Transportation Systems for the term 2004-2008; he served the IEEE Intelligent Transportation Systems Society as President for the term 2010-2011. He is recipient of two ERC (European Research Council) prestigious grants. Alberto is a IEEE Fellow.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

SoC for ultra HD mono and stereo-vision processing

Alberto Broggi
VisLab - Ambarella
abroggi@ambarella.com

Oct 1, 2018 – PPNIV, IROS, Madrid

VisLab

Group started in **mid '90s** at the University of Parma, Italy

Spin-off launched in **2009**

VisLab

Group started in **mid '90s** at the University of Parma, Italy

Spin-off launched in **2009**

Acquired by Ambarella in **2015**

Ambarella

- Chip company working on ultra-HD video

VisLab + Ambarella

- Starting July 2015 VisLab is working with Ambarella
  - Ambarella, a chip company
  - VisLab, a computer vision startup

Goal

- Design an engine for automotive systems (from ADAS to Autonomous Driving):
  - High performance
  - Low cost
  - Low power consumption
  - Automotive grade

to handle perception, data fusion, and ultimately also path planning

Ambarella CV SoC

- Current CV chip (CV-2):
  - 4k images (up to 8 image streams, incl multiscale) @30fps
  - IDSP on board, H.265 on board
  - Stereo processing @ 30fps (incl multiple stereo)
  - Monocular processing @ 30fps (CNNs, vector, serial)
  - Power consumption: under 5W
  - AEC-Q100

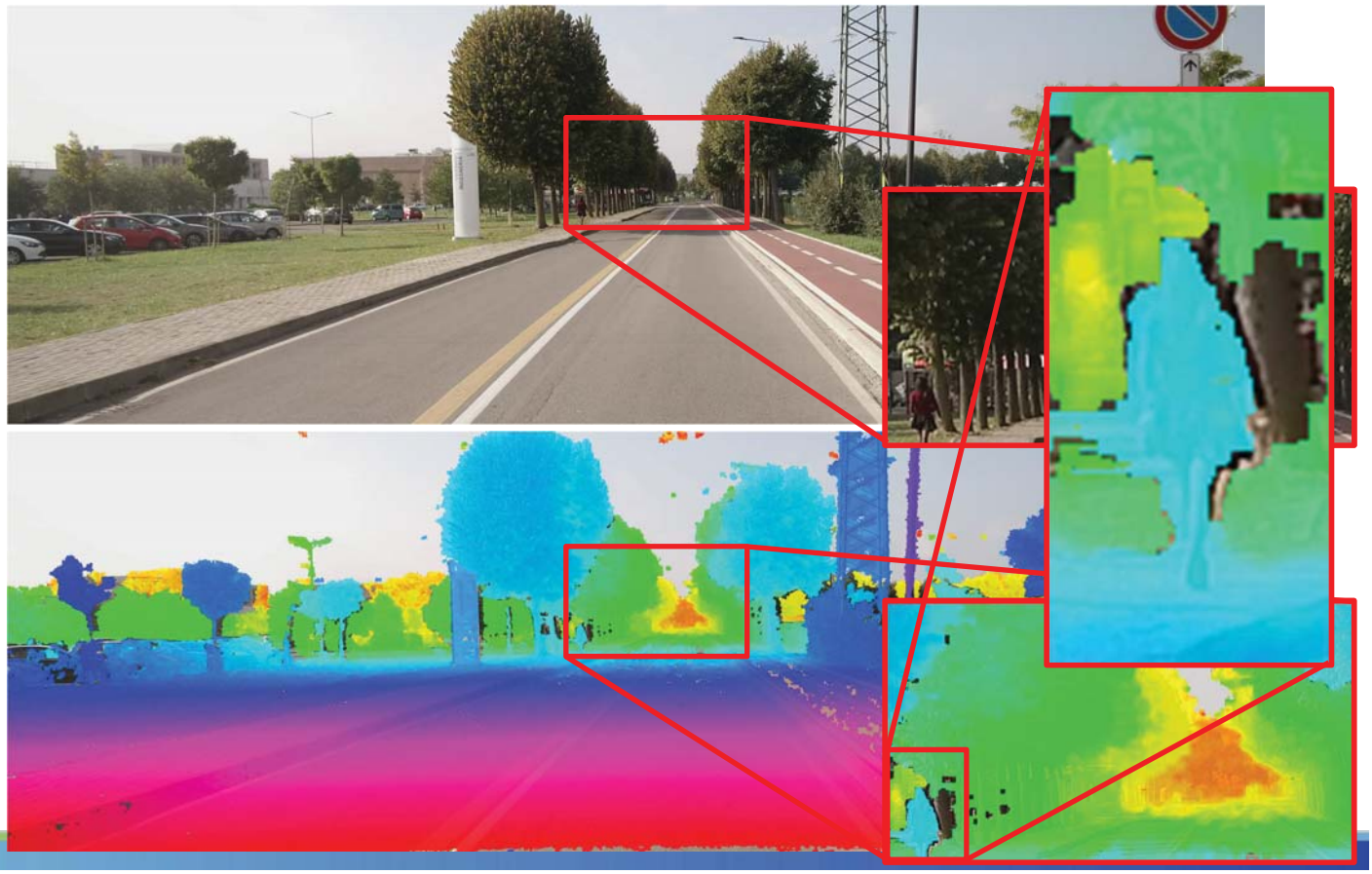
4k Image Resolution

4k, cropped (3840 x 1280)

Image Quality – Sun

Image Quality – Night

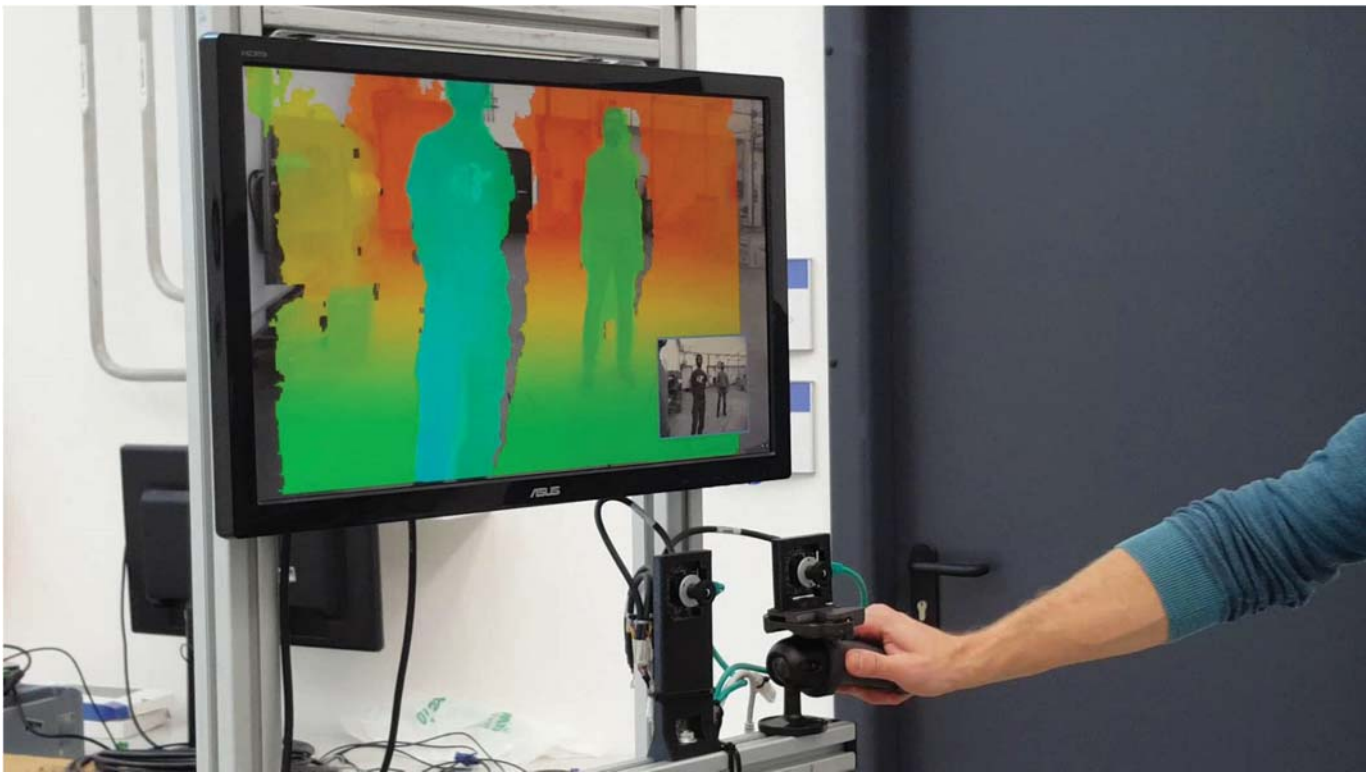
4k Stereo Vision



Stereo Calibration

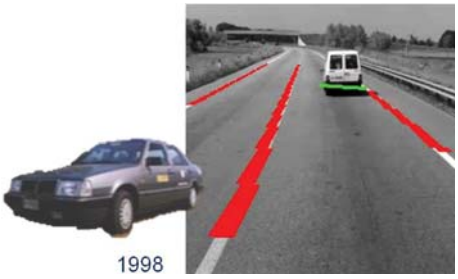
- In the past calibration has been one of the **major showstopper** for stereo vision, especially on vehicles
- A stereo camera is a measurement instrument
- Calibration needs to be maintained… for years

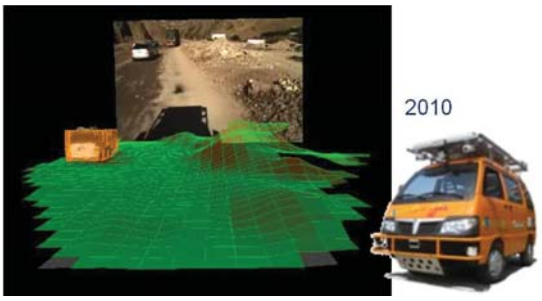
Stereo AutoCalibration

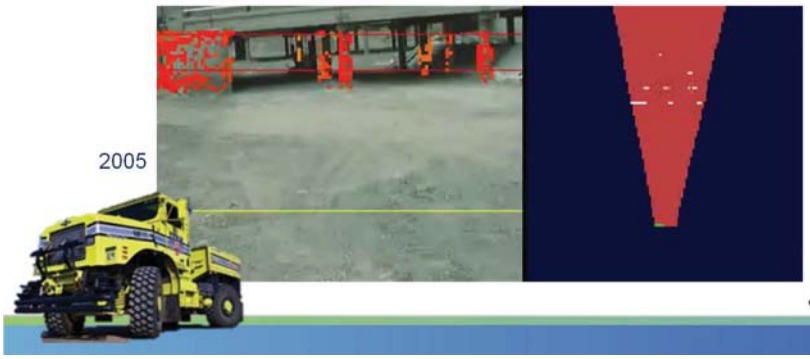


Stereo History

Stereo processing and autocalibration come from VisLab's multi-year history

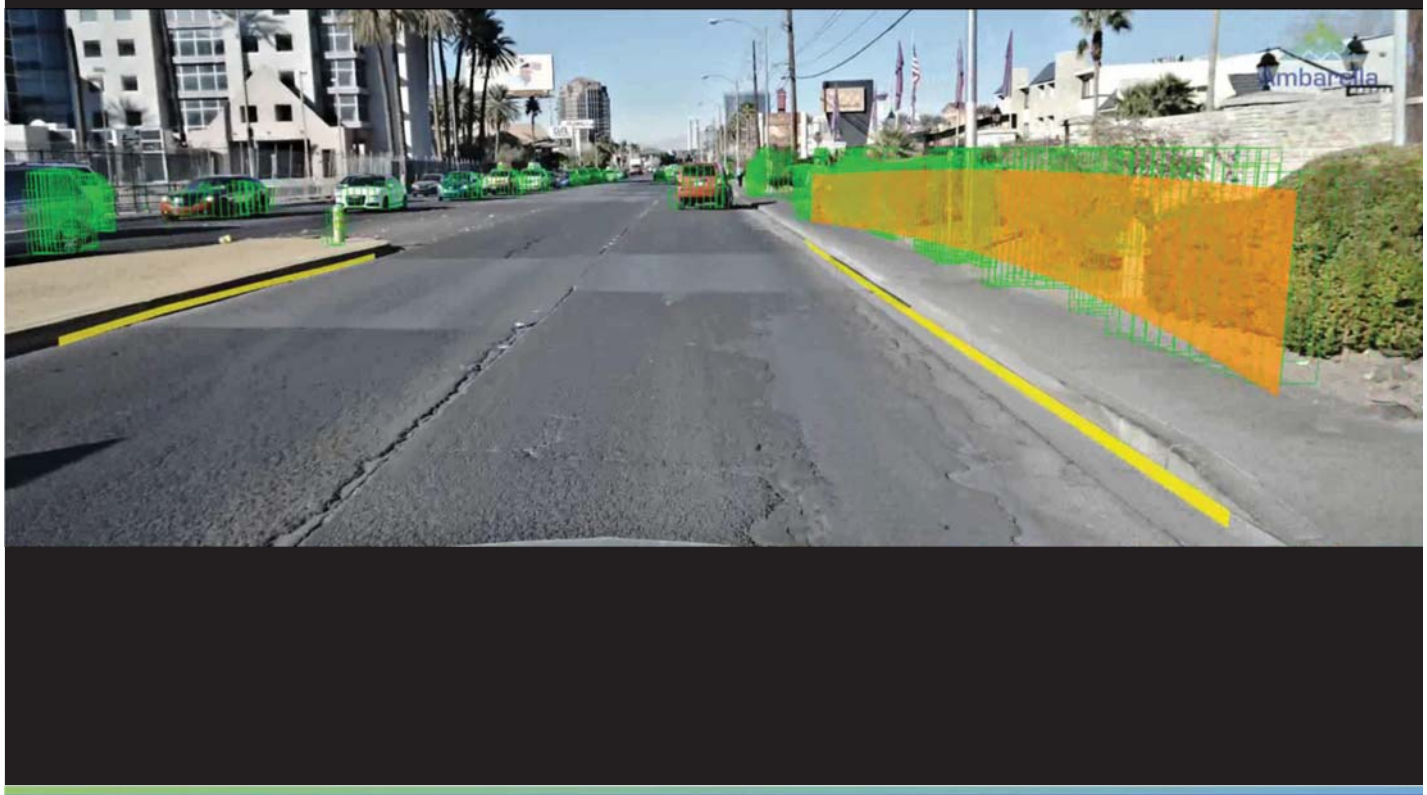






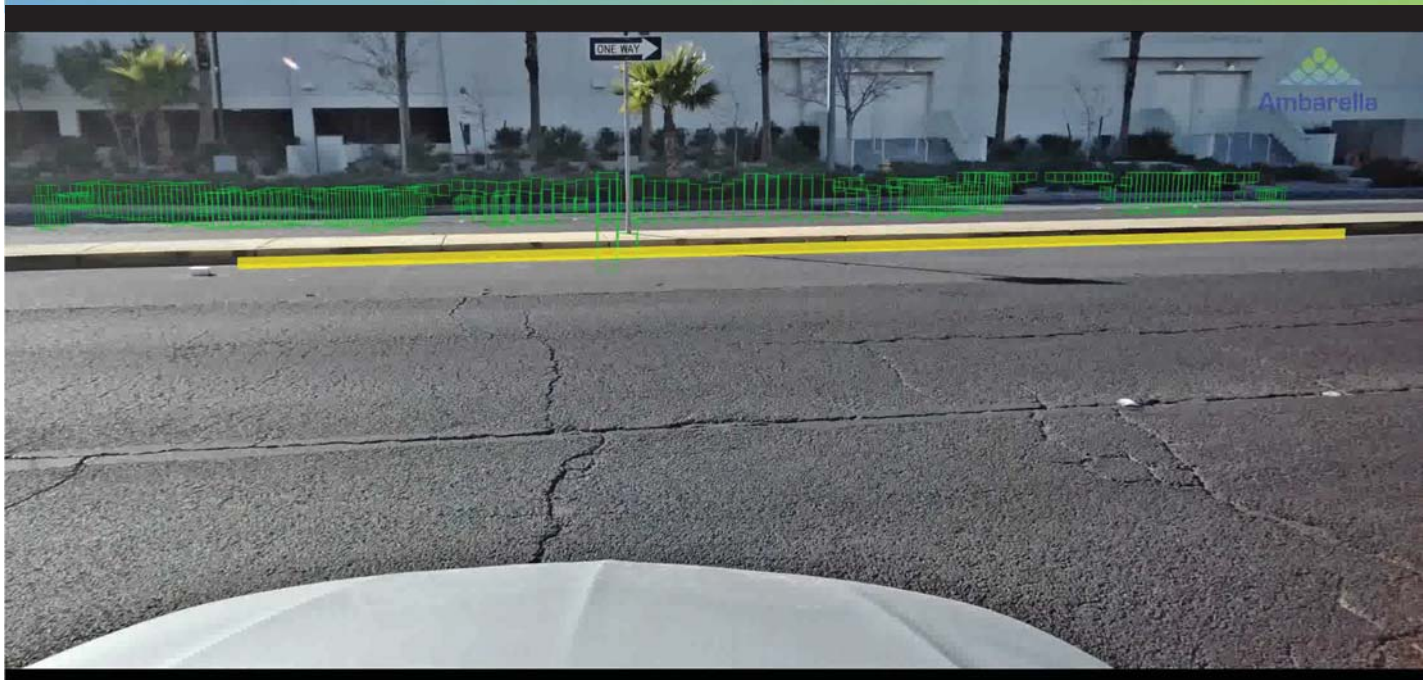
4k Stereo Vision

Ambarella™

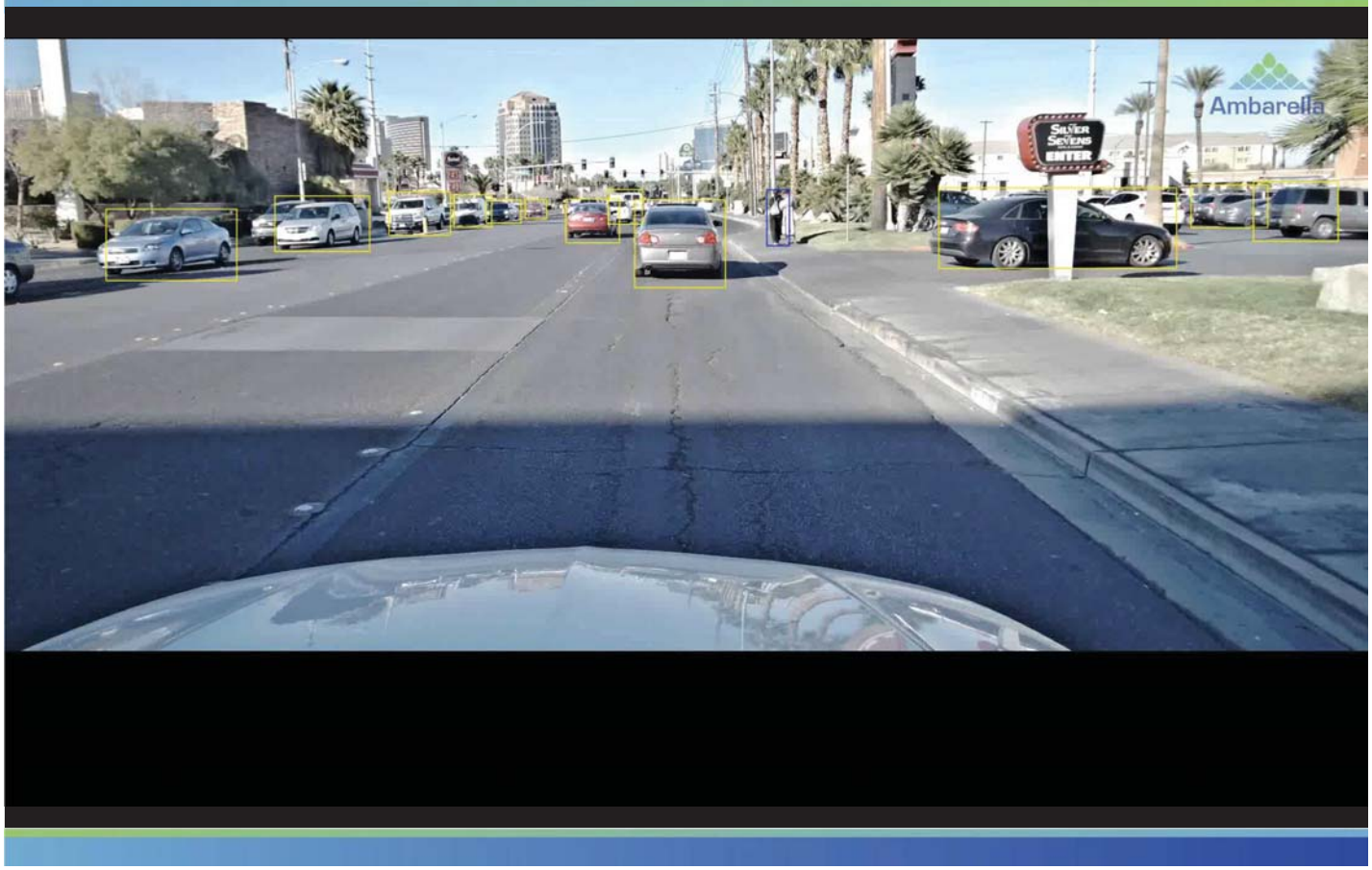


4k Stereo Vision

Ambarella™



4k CNN Classification

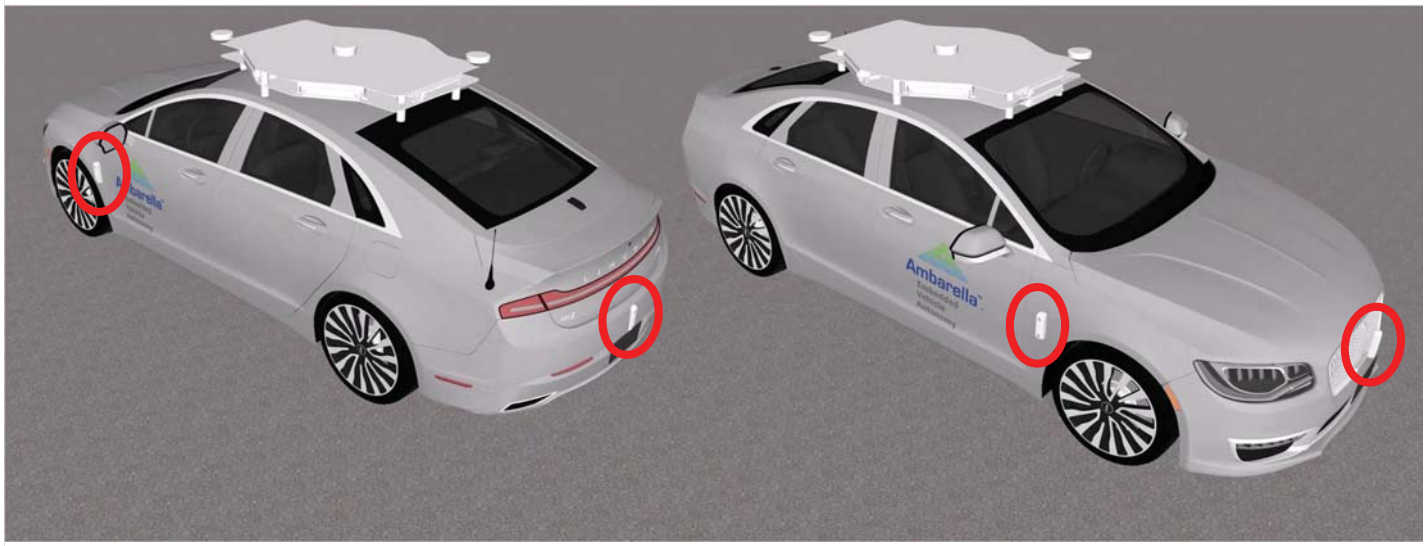


EVA – Embedded Vehicle Autonomy

EVA – Embedded Vehicle Autonomy

Ambarella™

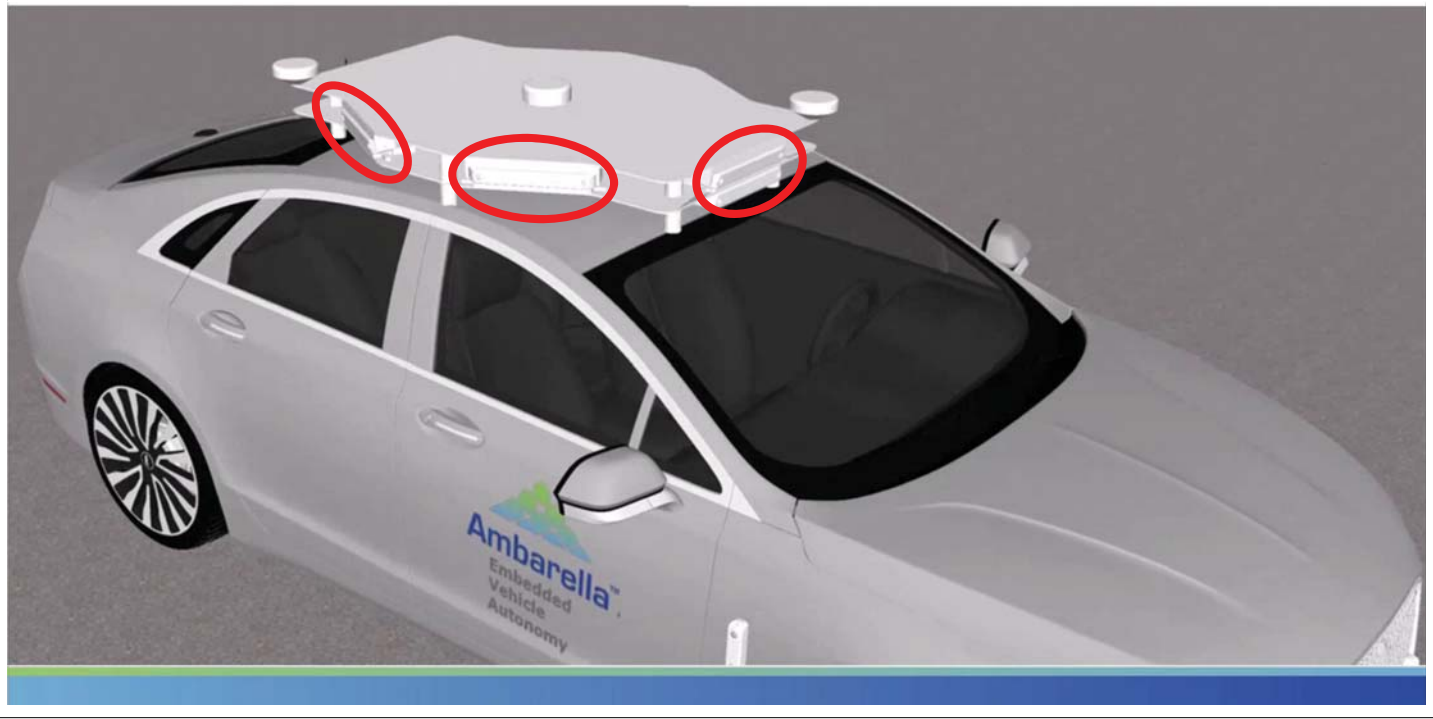
- Short Range Module: 4x 1080p stereo cameras



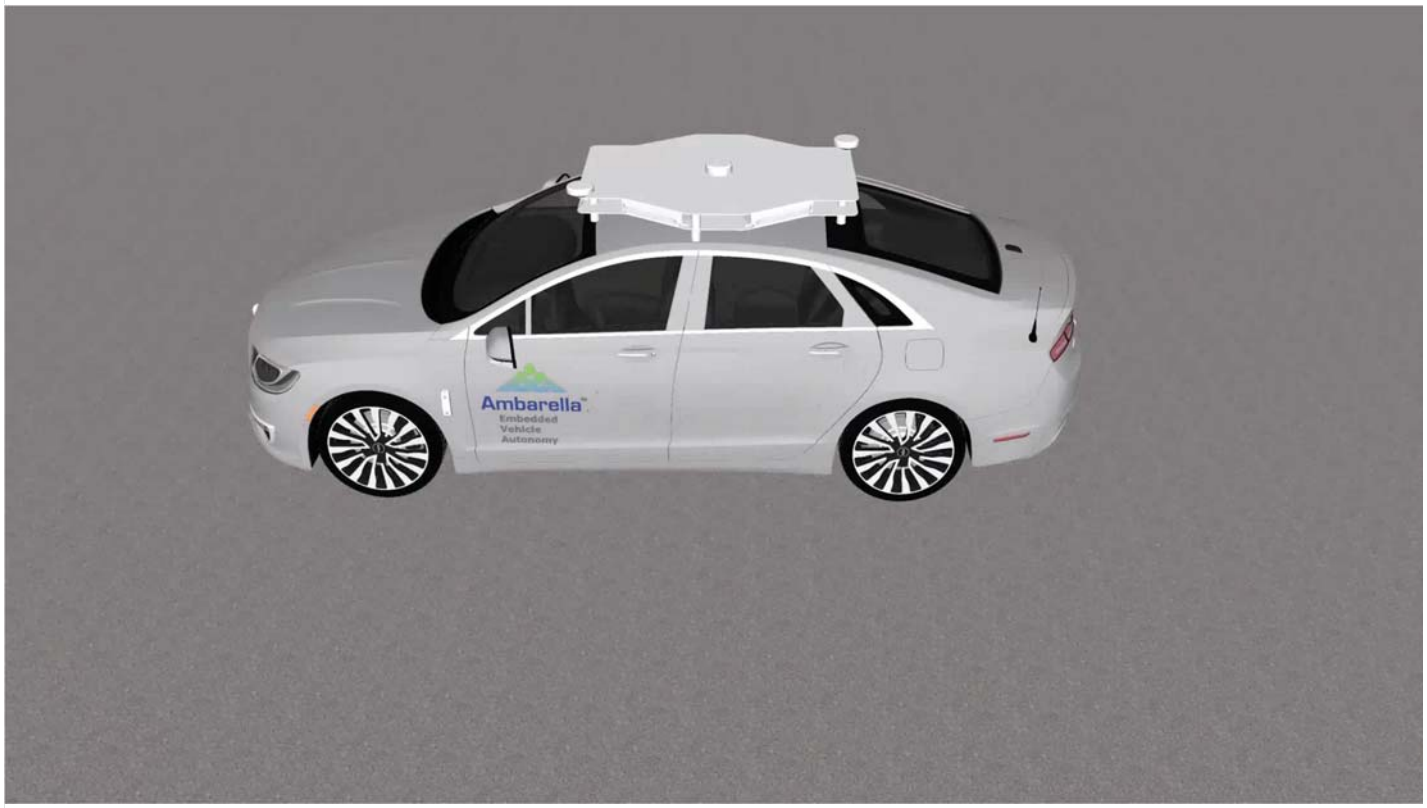
EVA – Embedded Vehicle Autonomy

Ambarella™

- Long Range Module: 6x 4k stereo cameras



EVA Stereo Vision Sensing



Conclusion

- Visual perception is key for intelligent vehicles
- We are porting advanced tools (like stereo and CNNs) into a low-cost, low-power, high performance chip
- The CV family: CV-1, CV-2, CV-22,…

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Perception

- **Title: Intelligent feature selection method for accurate laser-based mapping and localisation in self-driving**
  Authors: N. Hernandez, I. G. Daza, C. Salinas, I. Parra, J. Alonso, D. Fernandez-Llorca, M.A. Sotelo

- **Title: LiDAR based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape**
  Authors: E. Héry, P. Xu and P. Bonnifait

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Intelligent feature selection method for accurate laser-based mapping and localisation in self-driving cars

N. Hernández, I. G. Daza, C. Salinas, I. Parra, J. Alonso, D. Fernández-Llorca, M.Á. Sotelo, Fellow, IEEE
Computer Engineering Department
University of Alcalá, Madrid (Spain)
Email: noelia.hernandez@uah.es, ivan.garciad@uah.es, carlota.salinasmaldo@uah.es

***Abstract*—Robust 3D mapping has become an attractive field of research with direct application in the booming domain of self-driving cars. In this paper, we propose a new method for feature selection in laser-based point clouds with a view to achieving robust and accurate 3D mapping. The proposed method follows a double stage approach to map building. In a first stage, the method compensates the point cloud distortion using a rough estimation of the 3-DOF vehicle motion, given that range measurements are received at different times during continuous LIDAR motion. In a second stage, the 6-DOF motion is accurately estimated and the point cloud is registered using a combination of distinctive point cloud features. We show and analyse the results obtained after testing the proposed method with a dataset collected in our own experiments on the Campus of the University of Alcalá (Spain) using the DRIVERTIVE vehicle equipped with a Velodyne-32 sensor. In addition, we evaluate the robustness and accuracy of the method for laser-based localisation in a self-driving application.**

I. Introduction and related work

The booming field of self-driving cars has ushered in a new period of development in a number of scientific areas, being map building one of the most outstanding ones. A great deal of automotive companies are putting significant amounts of effort on building accurate 2D maps for automated driving purpose. Those maps contain information regarding the geo-referenced position and geometrical configuration of elements such as intersections, lane markers, road signs, road signals, etc. However, in order to achieve accurate localisation, self-driving cars need not only 2D maps but also 3D maps providing a distinct representation of the environment. Although some researchers have demonstrated the feasibility of using vision-only features for accurate localisation, such as the Daimler-KIT group did in the BERTHA route in Germany [1], 3D maps for accurate localisation are usually built using point clouds obtained with laser sensors. This is the case of Waymo [2] (formerly Google) self-driving cars, which have been performing automated driving missions in the Mountain View area in California for almost one decade already. A similar laser-based localisation approach has been followed by many researcher groups in the area of automated driving, such as the National Seoul University [3] or the SMART program (Singapore-MIT Alliance for Research and Technology) [4].

Fig. 1. DRIVERTIVE vehicle.

When it comes to automated driving, there is currently a debate in the scientific community regarding the trade-off between local perception capability and map dependence. It seems that further effort on the first topic is definitely needed, since self-driving cars have to make progress in their capability to better understand the world they see, very much in an attempt to mimic human driving style. However, map-based localisation is still a crucial, and necessary element in todays automated driving systems. In this regard, the use of laser for accurate localisation provides a number of advantages with respect to other sensors, such as vision. It is well known that vision-based mapping and localisation is prone to failure at night-time, in low visibility conditions or under adverse weather conditions. Change of appearance is another relevant problem. For example, a road diversion can provoke a failure in a vision-based localisation system if the new features that the car is finding as it moves have not been previously stored in the system. A similar problem can occur during the fall season, when the leaves of trees can fall down and derive in a situation of strong change of appearance with respect to the time when the map was built (if it was built during the spring or summer time).

However, the use of LIDARs for mapping and localisation does not come without difficulties. Thus, motion estimation via moving LIDARs involves motion distortion in point clouds, as range measurements are received at different times

during continuous LIDAR motion. Hence, the motion often has to be solved using a large number of variables and a computationally heavy optimization algorithm [5]. Scan matching also fails in degenerate scenes, such as those dominated by planar areas. Similarly, the localisation method can fail in situations in which a repetitive pattern is encountered, e.g. a park with symmetrically located trees, or in cases in which the amount of moving objects is largely predominant over the number of features provided by static elements. These difficulties require a further effort from the scientific community in order to develop really robust and fully operational laser-based mapping and localisation techniques for self-driving cars. In this line, Rohde [6] proposes a localisation method specifically designed to handle inconsistencies between map material and sensor measurements. This is achieved by means of a robust map matching procedure based on the Fourier-Mellin transformation (FMT) for global vehicle pose estimation. Consistency checks are then implemented for localisation integrity monitoring, leading to significantly increased pose estimation accuracy. Other approaches [7] segment moving parts in the sequence of point clouds, being capable of distinguishing rigid motions in dense point clouds and cluster those by applying segmentation schemes, such as graph-cuts into the ICP (Iterative Closest Point) algorithm. Consequently, the clustering process aims at segmenting moving vehicles out from the point cloud in an attempt to improve the accuracy of the laser-based localisation scheme. In [8], a vision and laser-based odometry system is presented in which an intelligent pre-selection of point-cloud features is carried out, using an appropriate distribution of edges and planes, with a view to increase the accuracy of the map while removing outliers.

In this paper, we propose a new method for feature selection in laser-based point clouds with a view to achieving robust and accurate 3D mapping. Pre-selection of features is absolutely necessary in order to achieve accurate map making capacity. Otherwise, the map incorporates artifacts that derive in loss of accuracy and, consequently, lack of localisation precision. The proposed method follows a double stage approach to map building. In a first stage, the method compensates the point cloud distortion using a rough estimation of the 3-DOF vehicle motion, given that range measurements are received at different times during continuous LIDAR motion. In a second stage, the 6-DOF motion is accurately estimated and the point cloud is registered using a combination of distinctive point cloud features. The appropriate combination of such features, reveals to be a powerful tool to achieving accurate mapping and robustness to aggressive motion and temporary low density of features. The proposed selection method has the potential to be applied both at the mapping and at localisation stages, leading to a significant improvement in terms of accuracy. We show and analyze the results obtained after testing the proposed method with a dataset collected in our own experiments on the Campus of the University of Alcalá (Spain) using the DRIVERTIVE vehicle equipped with a Velodyne-32 sensor [9]. In addition, we evaluate the robustness and accuracy of the method for laser-based localisation in a self-driving application.

The rest of the paper is organized as follows. Section II provides a description of the mapping algorithm. In section III, a revision of the localisation method is carried out. Section IV presents and discusses the experimental results attained with the DRIVERTIVE automated car. Finally, section V analyzes the main conclusions and future work.

II. Mapping

Outdoor scenes are characterized by being composed of objects which are placed within a wide range of distances. While mapping, small angular errors in the order of milliradians could lead to 1 meter errors for an object located at 50 meters. In addition, the point cloud is distorted by the vehicle egomotion during the acquisition. In order to achieve robust and accurate 3D mapping, a correction procedure of the 3D point clouds deformation is mandatory previous to the map creation.

A. 3-DOF sweep correction

Our experimental platform, DRIVERTIVE, consists on a commercial Citroën C4 modified for automated driving. DRIVERTIVE GPS-based localisation combines the information from an RTK-GPS, CAN bus and a low-cost IMU (Inertial Measurement Unit) in an 3-DOF EKF (Extended Kalman Filter) as explained in [9]. A Velodyne-32 sensor was attached approximately 50 cm over the vehicle's roof to perform the mapping and localisation tasks based on LIDAR odometry. The final purpose is to provide an accurate and robust localisation not based on RTK-GPS which suffers from undesired blackouts due to urban canyons, tunnels, trees, etc.

Our Velodyne-32 delivers approximately 10 sweeps of 360 degrees per second. This means that, at normal driving speeds, the point cloud sustains considerable deformation. To correct this deformation, the angular and linear velocities provided by the EKF are considered constant between two consecutive Velodyne-32 firings. The motion undergone by the Velodyne-32 between two consecutive firings is compensated to create a single pose reference for a 360 degree sweep (Fig. 2). It is worth noticing that, as roll and pitch angles are not taken into account, this initial estimation introduces errors, specially during sharp turns such as roundabouts and speed bumps.

This sweep correction is used as an initial rough guess for the next stage.

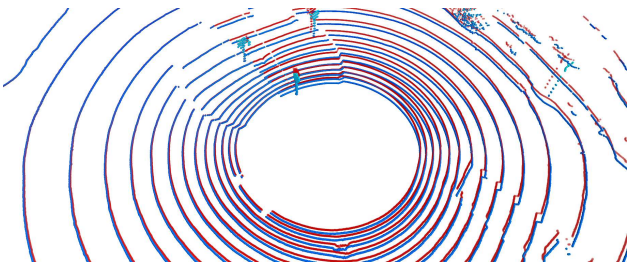


Fig. 2. 3-DOF sweep correction. In blue, the corrected Velodyne-32 sweep. In red, the original sweep.

B. 6-DOF LIDAR odometry

Using the 3-DOF initial guess, a registration technique will estimate the 6-DOF motion undergone by the vehicle to create a final corrected point cloud. This cloud will be used as input for an octomap-based mapping technique. For the registration process, distinctive point cloud features are extracted and selected using K-means and RANSAC. Then, an ICP will estimate the 6-DOF transformation on the features that will be used for the correction of the point clouds. Finally, the corrected point clouds will be used to create a 3D octomap-based representation of the environment.

Fig. 3 shows a block diagram of the algorithm.

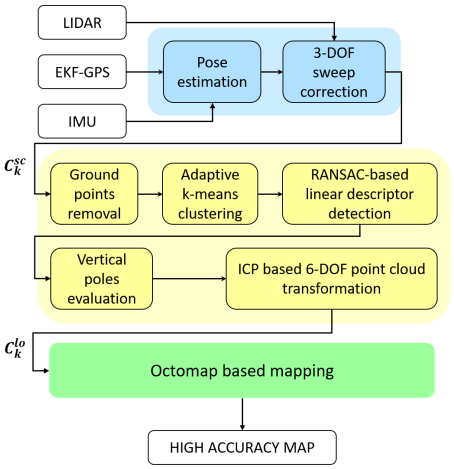


Fig. 3. Block diagram of the LIDAR odometry.

1) Linear descriptors extraction: Let us define $C_k^{sc} = \{p_1^{sc}, p_2^{sc}, \ldots, p_N^{sc}\}$ as the corrected point cloud using the 3-DOF sweep correction at time k composed of N world referenced 3D points $p_i^{sc} = \{x_i, y_i, z_i\}$.

Initially, a filtering process is applied to C_k^{sc} to remove ground points. This will help the clustering process in the next step. Next, an iterative adaptive k-means algorithm is applied to the filtered cloud to extract cluster candidates for the following step. The L2 Euclidean distance used on each k-means iteration is adapted to account for the sparseness of far objects. In a similar process to [10], the best fitting linear descriptor is computed using RANSAC for each one of the clusters. Only descriptors with a fitting value above a predefined threshold and with an orientation and size corresponding to a vertical pole are selected. Finally, these descriptors will be used in an ICP to estimate the 6-DOF transformation undergone by the descriptors.

2) 6-DOF LIDAR odometry estimation: An ICP procedure will estimate the 6-DOF transformation on the descriptors based on the extracted vertical descriptors. Two sets of descriptors, new ones and tracked ones, are needed for the ICP computation. The new set is composed of the vertical descriptors detected at the current point cloud. The tracked set is composed of the vertical descriptors tracked or matched in the previous ICP iteration. Once the ICP transformation is computed, the vertical descriptors are matched in a brute force search analyzing the euclidean distance between them. Those closer to a predefined threshold will be added to the tracked set.

The ICP obtains the linear and angular transformation needed to reach the minimum matching error. This transformation is only applied when two or more descriptors are available in both the new and the tracked descriptors sets. Otherwise, the ICP geometrical transformation will provide inaccurate results.

Algorithm 1: LIDAR odometry correction

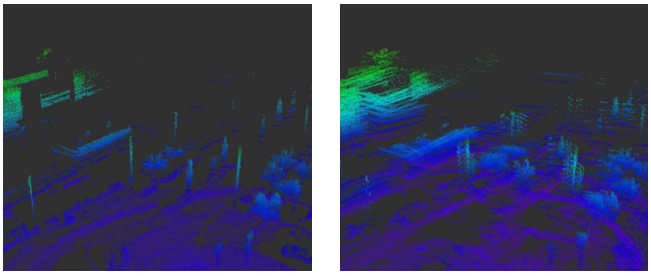
Input: C_k^{sc} is the point cloud with a sweep correction.
Result: C_k^{lo} is the point cloud with 6-DOF LIDAR odometry correction.
Data: $S \longleftarrow \emptyset$ is the set of segmented point clouds.
Data: $D \longleftarrow \emptyset$ is the set of linear descriptors at current time. Where $D_i = [x_i, y_i, z_i, \vec{x}_i, \vec{y}_i, \vec{z}_i]$.
Data: D^{map} is the set of linear descriptors in the mapped environment.
Data: $D_{target} \longleftarrow \emptyset$ is the set of target descriptors that will be used by the ICP procedure.

```
begin
    for x = {1, ..., 5} L2 Euclidean distance values. do
        S ⟵ EuclidianCluster(x, C_k^SW)
    for S_i ∈ S do
        D_aux ⟵ RANSAC(inliers_threshold = 1m, S_i) Auxiliary descriptor.
        if z_i > 0.9997 then
            D ⟵ D_aux
    if First_iteration then
        D^map ⟵ D
    else
        for i ∈ len(D) do
            for j ∈ len(D^map) do
                if EucliDistance(D_i, D_j^map) > 1m then
                    D^map ⟵ D_i
                else
                    D_source ⟵ D_i
                    D_target ⟵ D_j^map
        6DOF_correction ⟵ ICP(D_source, D_target)
        C_k^lo ⟵ 6DOF_correction · C_k^sc
```

C. Map creation

The geometrical transformation obtained in the previous step is now used over the C_k^{sc} point cloud. This way, a new corrected C_k^{lo} point cloud is obtained using the 6-DOF LIDAR odometry correction.

A high-resolution 3D map will be created using this new point cloud as the input for an Octomap algorithm [11]. Fig. 4 depicts the effect of the cloud correction on the map creation.



(a) (b)

Fig. 4. Results of mapping with and without correction. (a) Vertical poles with correction. (b) Vertical poles without correction.

III. Localisation

Indoor localisation has been traditionally solved by the use of 2D map representations and 2D sensors. This approach, which has been proved reliable and accurate indoors, is not enough to provide accurate outdoor localisation for autonomous vehicles. As a consequence, our objective is to obtain high definition 3D maps that are suitable to perform outdoor localisation.

In this paper, we propose to use a Monte Carlo Localisation method (MCL) [12], also called Particle Filter (PF), adapted to use high definition 3D maps and measurements collected from a high definition 3D sensor (Velodyne-32 LIDAR). A similar approach has been previously used in [13] using 3D maps and 2D sensors to perform indoor localisation. In this method a PF is used to obtain the 6D pose (3D position (x, y, z) and the roll, pitch and yaw angles (φ, θ, ψ)) of a humanoid robot carrying a Hokuyo laser on the head.

For the autonomous vehicle localisation we adapted these methods to obtain a 3D pose (2D position (x, y) and the yaw angle (ψ)) using a ray casting model to evaluate the fitness of the point cloud over the 3D map. The main idea behind this method is to keep a set of particles that will represent possible locations of the vehicle. Each particle is scored according to the similarity between the real measurements collected using the Velodyne-32 and the measurements that should be obtained provided the vehicle was located exactly on a particle pose. Finally, the vehicle's location can be estimated using the pose of the particles with the highest weights. This method consists of the following steps:

A. Initialisation

During the initialization step, particles have to be distributed over the map covering all possible vehicle poses. This area could be thousand square meters in outdoor environments making unfeasible the initial distribution of the particles over the whole map. Therefore, the initial distribution must be reduced to the poses around the initial position of the vehicle. This position is obtained using the rough location provided by a Garmin 18x LVC GPS (accuracy <15 meters, 95% typical), not enough for autonomous vehicles navigation but enough to initialise the filter. In order to reduce the initial error, a fixed number of equally weighted particles are randomly generated covering the area around the initial position.

B. Update (Weight computation)

At this step, a weight will be computed for each particle containing the probability of being the real location of the vehicle. The weight of each particle will be computed by scoring the similarity between the point cloud collected using the Velodyne-32 and the measurements that should be obtained provided the vehicle was located exactly on a particle pose. This similarity will be scored by using a ray casting algorithm computed from the position of each particle (Fig. 5). This way, one beam will be launched in the direction of each point p_i in the point cloud until it intersects with an object on the map. Then, a score $\phi(p_i)$ (Eq. 1) will be calculated depending on the difference d between the distance to the real point d_r and the distance to the intersection with the map d_m. An additional score is added if d_r is smaller than d_m (to cover occlusions, highly likely in driving outdoor scenes).

$$\phi(p_i) = \begin{cases} \alpha \exp\left(-\frac{d^2}{2\sigma^2}\right) & , d_r \geq d_m \\ \alpha \exp\left(-\frac{d^2}{2\sigma^2}\right) + \beta \left(\frac{\exp(\lambda d_r)}{1-\exp(\lambda d_m)}\right) & , d_r < d_m \end{cases} \quad (1)$$

, where σ is the sensor noise and α, β, and λ are weighting factors.

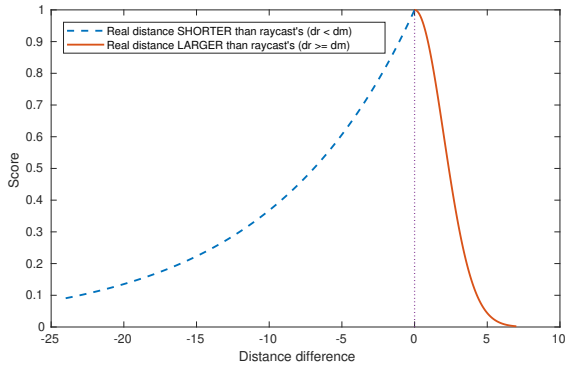


Fig. 5. Score given to a point depending on the difference between the real distance and the distance to the intersection on the map.

The score Φ assigned to each particle will be the sum of the scores of each point in the point cloud $\Phi = \sum \phi(p_i)$. Finally, the particle weights are normalised so they sum up 1.

The point cloud is filtered to reduce the computational effort required to calculate the weights of the particles. To do so, an intelligent feature selection is applied: by eliminating points close to each other (with distances under the map resolution), by removing the points on the ground (there are no differential features on the ground plane) and selecting points on distinctive features such as corners or poles.

C. Pose estimation, resampling and propagation

Once the weight for all the particles is computed, the most likely pose of the vehicle is estimated as the mean pose of the particles with the highest weights. Then, during the resampling stage, the particles with high weights are replicated while the particles with low weights are removed to avoid the degeneration of the particle cloud. Finally, the particles are propagated using the vehicle's motion model to continue with the next iteration of the PF.

IV. Experimental Analysis

The final objective of creating a high quality map is to obtain accurate localisation of the vehicle. However, there are many factors to take into account in order to evaluate the map creation accuracy and the localisation performance.

Firstly, in our system, an RTK-GPS-based EKF is used as groundtruth, but it is not free of errors as we will show in the next sections. This implies that corrections made in the mapping or localisation phases over the EKF data will be considered as errors in the final results. As a consequence, and to give a more realistic figure of the mapping accuracy, street poles positions were manually measured using an RTK-GPS (Fig. 6). These poles positions were averaged over 500 samples. The mean distance and variance of the mapped poles positions to the real ones will be used as an indication of the mapping accuracy.

Secondly, the numeric results of the localisation stage should be considered as tentative and will have to be validated on autonomous driving experiments where the localisation outputs will be used for navigation tasks.

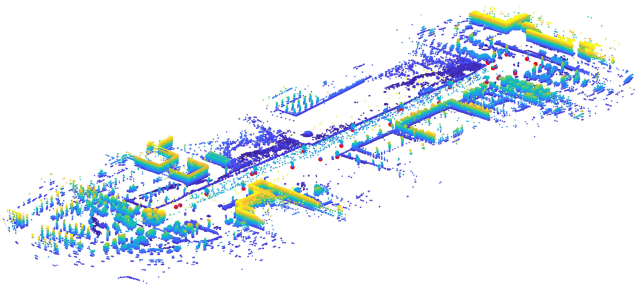
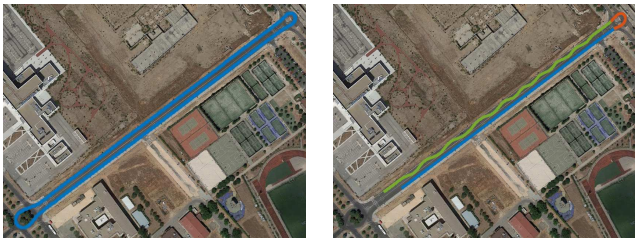


Fig. 6. Poles position groudtruth represented as red dots.

A. Experimental set-up

The experiments were performed on the Universidad de Alcalá (UAH) campus located at Alcalá de Henares (Madrid, Spain). The test area is a semi-industrial compound with wide open areas and large buildings connected by roundabouts. Data was collected driving in real traffic conditions for mapping and localisation on two consecutive runs (Fig. 7). Mapping data was collected at an approximate speed of 17 km/h in a naturalistic driving (Fig. 7(a)), while localisation test data was collected, first on a straight line, and then swerving (Fig. 7(b)).

Finally, the groundtruth was obtained using a 3-DOF EKF (based on RTK-GPS information) as explained in section II-A. For the localization stage, the mean euclidean distance error between the PF estimation and the groundtruth is used as performance indicator.



(a) (b)

Fig. 7. Experimental environment and trajectories. (a) Trajectory for map creation. (b) Trajectory for localisation tests. In blue, straight line, in red roundabout and in green swerving.

B. Mapping results

A reference map, based only on the EKF positions and the pointcloud without any further pre-processing, was created to compare the mapping results. In the creation of this map, the errors observed in the EKF positioning due to loss of coverage or loss of corrections were manually removed. The idea was to establish a baseline for comparison. It is worth reminding that the EKF estimation does not include pitch and roll angles, and thus, some improvement was expected to be gained with the 6-DOF LIDAR odometry correction. Table I shows the mean distance and variance of the poles position for both maps.

TABLE I
Mean Euclidean Distance Error and variance of the poles positions

	Mean Euclidean Distance Error	Variance
Reference map	22.30 cm	6.00 cm
6-DOF LIDAR odometry	9.74 cm	4.16 cm

As expected, the 6-DOF LIDAR odometry mapping reconstructs more accurately the poles position by a factor of almost two. This shows that the 6-DOF LIDAR odometry technique is able to correct for some of the errors introduced by the EKF and that pitch and roll angles estimation have a significant effect in the map creation accuracy.

C. Localisation results

The localisation system described in Section III was tested on both, the "error free" reference map and the 6-DOF LIDAR odometry based one. Our purpose was two-fold: First, to test the effect on localisation of introducing corrections of pitch and roll angles on the map creation. Second, to evaluate the performance of a map created relying on LIDAR odometry. Table II shows the mean localisation distance errors for both, the reference and the 6-DOF LIDAR odometry maps in the test trajectory.

TABLE II
MEAN LOCALIZATION DISTANCE ERROR AND VARIANCE (CM)

	Lateral	Longitudinal	Total
Reference map	14.86 ± 0.89	20.65 ± 5.23	28.27 ± 4.60
6-DOF LIDAR	13.63 ± 1.85	21.13 ± 2.42	27.83 ± 2.85

The performance of the localisation on the 6-DOF LIDAR odometry map is comparable to the "error free" reference map indicating that is possible achieve similar levels of accuracy using LIDAR odometry instead of RTK-GPS. Fig. 8 shows an example where the PF is correctly estimating the vehicle's position, but the EKF groundtruth is off by about 1 metre. As explained before, some of the localisation error is accounted for EKF errors caused by RTK-GPS blackouts, meaning that the final localisation accuracy should be slightly higher.

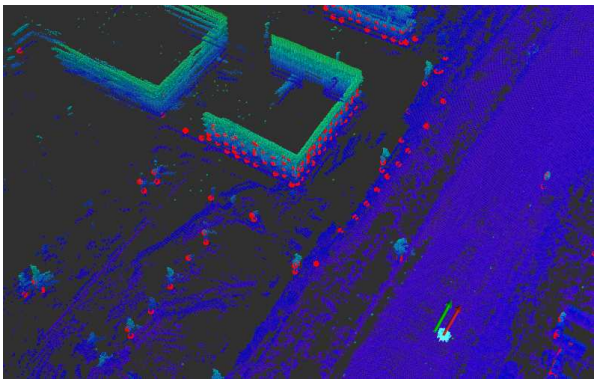


Fig. 8. PF (red arrow) and EKF groundtruth (green arrow) localisation results. The Velodyne-32 hits are represented as red dots.

Although more environments and challenging situations (i.e. strong occlusions) should be tested, these preliminary results indicate that the mapping and localisation techniques are accurate enough for navigation tasks in autonomous vehicles. This method is a first approach towards the removal of RTK-GPS from the mapping and localisation stages.

V. CONCLUSION

In this paper, we proposed a new method for feature selection in laser-based point clouds with a view to achieving robust and accurate 3D mapping. The proposed method follows a double stage approach to map building: a 3-DOF point cloud distortion compensation and a 6-DOF LIDAR odometry-based motion estimation. Experiments were performed while driving in real traffic conditions in a semi-industrial compound. The results show that our mapping technique increases the mapping accuracy by a factor of two, while maintaing performance on the localisation stage. Our approach is a first step towards the full removal of RTK-GPS from both mapping and localisation stages.

As future work, for the localisation stage, we plan to test on different environments with strong occlusions. For the mapping stage, we want to introduce additional linear features and to pitch the Velodyne-32 around 45° only for the map creation. This is expected to reduce sparseness of the maps and also some of the errors introduced by the furthest targets.

ACKNOWLEDGMENT

This work was funded by Research Grants SEGVAUTO S2013/MIT-2713 (CAM), DPI2017-90035-R (Spanish Min. of Economy), and BRAVE, H2020, EC Contract #723021. This project has also received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737469 (AutoDrive Project). This Joint Undertaking receives support from the European Union Horizon 2020 research and innovation programme and Germany, Austria, Spain, Italy, Latvia, Belgium, Netherlands, Sweden, Finland, Lithuania, Czech Republic, Romania, Norway.

REFERENCES

[1] T. Dang, J. Ziegler, U. Franke, H. Lategahn, P. Bender, M. Schreiber, T. Strauss, N. Appenrodt, C. Keller, E. Kaus, C. Stiller, and R. Herrtwich, "Making bertha drive an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, 2014.

[2] "Google self-driving car project," accessed on April 2018. [Online]. Available: https://www.google.com/selfdrivingcar/

[3] S. Kim, G. Gwon, W. Hur, D. Hyeon, and S. Seo, "Autonomous campus mobility services using driverless taxi," *IEEE Intelligent Transportation Systems Magazine*, vol. 18, no. 12, 2017.

[4] "SMART: Singapore-MIT Alliance for Research and Technology," accessed on April 2018. [Online]. Available: https://smart.mit.edu/images/pdf/news/2013/SMART_Autonomous_Vehicle.pdf

[5] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "LiDAR point clouds correction acquired from a moving car based on CAN-bus data," *CoRR*, vol. abs/1706.05886, 2017. [Online]. Available: http://arxiv.org/abs/1706.05886

[6] J. Rohde, B. Volz, H. Mielenz, and J. M. Zollner, "Precise vehicle localization in dense urban environments," in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC 2016)*, 2016.

[7] Y. Kim, H. Lim, and S. C. Ahn, "Multi-body ICP: Motion segmentation of rigid objects on dense point clouds," in *Proceedings of the 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2015)*, 2015.

[8] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2015)*, 2015, pp. 2174–2181.

[9] I. P. Alonso, R. I. Gonzalo, J. Alonso, A. García-Morcillo, D. Fernández-Llorca, and M. A. Sotelo, "The experience of DRIVERTIVE-DRIVERless cooperaTIve VEhicle-Team in the 2016 GCDC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1322–1334, 2018.

[10] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS)*, 2014.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com

[12] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1999)*, vol. 2, 1999, pp. 1322–1328.

[13] A. Hornung, K. M. Wurm, and M. Bennewitz, "Humanoid robot localization in complex indoor environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010.

LiDAR based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape

Elwan Héry, Philippe Xu and Philippe Bonnifait
Sorbonne universités, Université de Technologie de Compiègne, CNRS UMR 7253 Heudiasyc
57 Av. Landshut CS 60319, 60203 Compiègne cedex, France
{elwan.hery - philippe.xu - philippe.bonnifait}@hds.utc.fr

***Abstract*— Relative localization between autonomous vehicles is an important issue for accurate cooperative localization. It is also essential for obstacle avoidance or platooning. Thanks to communication between vehicles, additional information, such as vehicle model and dimension, can be transmitted to facilitate this relative localization process. In this paper, we present and compare different algorithms to solve this problem based on LiDAR points and the pose and model communicated by another vehicle. The core part of the algorithm relies on iterative minimization tested with two methods and different model associations using point-to-point and point-to-line distances. This work compares the accuracy, the consistency and the number of iterations needed to converge for the different algorithms in different scenarios, e.g. straight lane, two lanes and curved lane driving.**

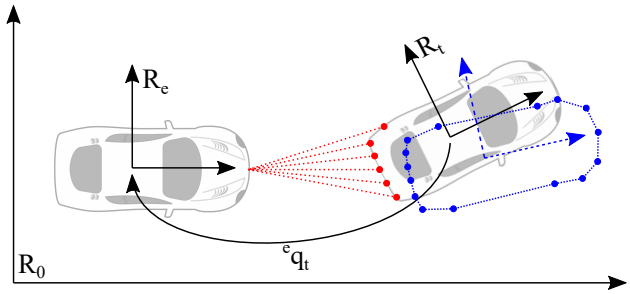


Fig. 1: *The ego-vehicle estimates the rigid transformation ${}^{e}q_t$ that maps the reference frame R_t of a target vehicle into its own reference frame R_e. The target vehicle communicates an estimate of its pose along with a polygonal model of its shape (illustrated in blue with an unavoidable error).*

I. Introduction

Vehicle detection and tracking are a key features for autonomous driving which have led to many research work [7]. Knowing the relative pose of a detected vehicle in the ego-vehicle reference frame is essential for tasks such as obstacle avoidance or platooning.

The emergence of wireless communication capabilities for vehicles in the recent years has given rise to new possibilities. Having access directly to information such as pose or vehicle dimensions, e.g., from the European standard CAM (Cooperative Awareness Message) [5], a vehicle can have a better understanding of its surroundings. Moreover, if a vehicle can receive perception information from other vehicles, it can have an augmented perception of the environment enabling it to see much further. However, in order to transpose perception information of one vehicle into the reference frame of another, the relative pose between these two vehicles is needed.

Many vision based vehicle detection algorithm can be found in the literature with recent deep learning based detectors having impressive performances [4]. However, these methods often only return a bounding box in the image frame and fail at providing a metric estimate of the pose of the detected vehicle. On the contrary, LiDAR based vehicle detection are much more adapted for relative pose estimation. Vehicle detection by fitting a geometrical model such as L-shape fitting [10] provides a good estimate of the relative pose. Because the true shape of a detected vehicle is not known a priori, only simple geometric models, i.e., box, are usually used for model fitting. However in the context of communicating vehicles, it is possible for a vehicle to send an accurate model of its own geometric shape.

An Iterative Closest Point (ICP) algorithm is often used for scan matching [3] but can also be used to fit a scan to a model [9]. ICP provides a good estimate of the relative pose but the associated covariance matrix is often not computed which is crucial to obtain an information that can be used in a data fusion process [2], [8], [1]. With that objective in mind, we present in this paper several algorithms for relative localization based on model matching with covariance estimation.

The paper is organized as follows. In Sec. II we introduce the relative localization problem. In Sec. III, the iterative minimization algorithm is presented with four different matching methods to associate LiDAR points to a geometric model. We also introduce two different minimization methods. Finally, the results of the different matching and minimization methods are compared in Sec. IV.

II. Problem statement

In this work, we aim to estimate the relative pose between two vehicles and quantify its uncertainty. We assume that an ego-vehicle, equipped with a LiDAR sensor, perceives a target vehicle resulting in set of 2D points $P = \{p_i = [x_i, y_i],\ i = 1, \dots, n\}$, in the ego-vehicle reference frame R_e. There exists many algorithms in the literature to compute this cluster of points from a LiDAR point cloud. This computation is out of the scope of this work. In our study,

we also suppose that the target vehicle communicates an estimate of the pose of its reference frame R_t along with a 2D polygonal model, $M = \{m_j = [x_j, y_j],\ j = 1, \dots, N\}$, representing its geometrical shape (see Fig. 1). The points m_j represent the vertices of the model and the edges are defined by two consecutive vertices $(m_j; m_{j+1})$.

This problem can be solved in 3D using a multilayer LiDAR and a 3D model with facets, e.g. STL model. Nevertheless a monolayer LiDAR is less expensive and gives already very good results with smaller computation time and information to communicate. The 2D hypothesis cannot always be respected, i.e. if the vehicles are not driving on a flat road or if the LiDAR scan and the 2D model are not on the same plane, e.g. when the model is at the height of the bumper of a truck and the LiDAR at the height of the bumper of a car. In these cases a 2D polygonal model can be computed from the intersection of a 3D model and the plane of the LiDAR scan.

The problem we aim at solving is to estimate the relative pose ${}^e q_t = [{}^e x_t, {}^e y_t, {}^e \theta_t]$ (θ being the heading) of the target vehicle in the reference frame of the ego-vehicle. In other words, ${}^e q_t$ represents the rigid transformation, i.e., translation and rotation, that maps R_t to R_e. Any point $p_t = [x_t, y_t]$ in R_t can be transformed into R_e as

$$\overline{p}_e = {}^e T_t \overline{p}_t = \begin{bmatrix} \cos({}^e\theta_t) & -\sin({}^e\theta_t) & {}^e x_t \\ \sin({}^e\theta_t) & \cos({}^e\theta_t) & {}^e y_t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} \quad (1)$$

where ${}^e T_t$ is the transformation matrix associated to ${}^e q_t$ and $\overline{p} = [p^T\ 1]^T$ is the homogeneous vector associated to p.

This problem can also be formulated as finding the transformation that would map the target model M to the perceived set of points p_i minimizing a positive scalar error E:

$${}^e\hat{q}_t = \arg\min_q E(q) = \arg\min_q \sum_{i=1}^{n} d(q; p_i, M), \quad (2)$$

where $d(q; p_i, M)$ represents a distance from point p_i to the model M corresponding to the transformation q.
By supposing that the minimization problem is convex, one can also compute the covariance matrix of the error by using an empirical estimate of the variance of the residuals, as proposed in [1]:

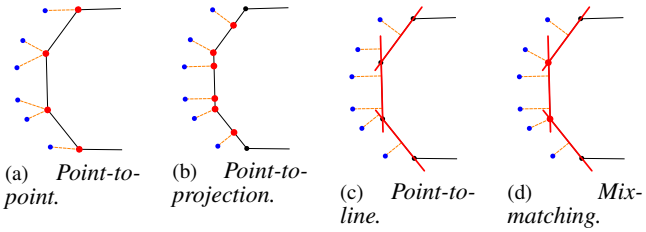
$${}^e\Sigma_t = 2\frac{E({}^e\hat{q}_t)}{n-k}\left(\frac{\partial^2 E}{\partial q^2}({}^e\hat{q}_t)\right)^{-1}, \quad (3)$$

where k is the dimension of ${}^e q_t$, i.e., $k = 3$. This computation needs at least four LiDAR points in the scan ($n \geq 4$).

In this paper, we compare several ways to compute the distance metric d, two different minimization methods within an Iterative Closest Point framework and we evaluate the consistencies of the estimated covariance matrices.

III. Computation of the Relative Pose

ICP is often used to match a LiDAR scan with another one. We used a similar method to estimate the relative pose between the model of the target vehicle and the scan. The objective is to find the pose that minimizes the error between the scan and the shape model.



(a) *Point-to-point.* (b) *Point-to-projection.* (c) *Point-to-line.* (d) *Mix-matching.*

Fig. 2: *Matching between the LiDAR points (in blue) and the model (in black). The matched points and lines are illustrated in red.*

Algorithm 1 *Overview of the proposed method.*

1: Compute a first rough relative pose using bounding boxes of the scan and of the received shape model
Loop
2: Match the clustered scan points with the model
3: Find the pose that minimizes the error
4: Break if the variation of the error divided by the number of LiDAR points is smaller than a threshold
End loop
5: Compute the covariance matrix

Algorithm 1 summarizes the method. In the following, we study four matching methods and two minimization strategies.

A. LiDAR points to model matching

We introduce four different ways to match a set of LiDAR points to a polygonal model.

- Point-to-point, ICP (Fig. 2a): each LiDAR point is matched with the closest vertices of the model. This method does not take into account the edges of the model. A sparsely discretized model may lead to a large distance between the LiDAR points and the model.
- Point-to-projection, ICPP (Fig. 2b): to have a tighter matching, one can match a LiDAR point to the nearest point of the model considering both its vertices and edges. This point can be projected onto the model using the smallest distance : the orthogonal distance to an edge or the Euclidean distance to a vertex.
- Point-to-line, PLICP (Fig. 2c): by matching a point to its orthogonal projection may result in an increase of the error since the matched point remains fixed during the minimization. One way to take this into account is to match the point directly to the line defined by its edge matched using the point-to-projection approach.
- Mix-matching, mixICP (Fig. 2d): the last method is a mix-matching using point-to-point matching when the smallest distance to the model is an Euclidean distance to a vertex and point-to-line matching when it is an orthogonal distance to an edge.

The difference between mixICP and ICPP is subtle. With point-matching methods the matched points are constant

whereas the scan can slide along the model with line-matching.

In the case of a matching between a LiDAR point p_i and a model point m_j (a vertex or an orthogonal projection):

$$d(M, p_i; q) = \|Tm_j - p_i\|^2 = \left\|m_j - T^{-1}p_i\right\|^2, \quad (4)$$

where the T is the transformation matrix associated to q. It should be noted that fitting the model points m_j to the LiDAR points p_i using T is equivalent to fit p_i to m_j using T^{-1}.

In the case of a matching between a LiDAR point p_i and an edge $(m_j; m_{j+1})$ with a unit normal n_j:

$$d(M, p_i; q) = \left(\left(m_j - T^{-1}p_i\right) \cdot n_j\right)^2. \quad (5)$$

B. Minimization using polynomial roots

The error function (2) to minimize is non linear. Censi [3] proposed to change the variable $q = [x\, y\, \theta]^T$ to $q_{4D} = [x\, y\, c\, s]^T = [x\, y\, \cos\theta\, \sin\theta]^T$. By doing so, the minimization of (2) using the distances (4) or (5) can be rewritten as a constrained quadratic problem:

$$\begin{cases} \min_{q_{4D}} & E(q_{4D}) = q_{4D}^T A q_{4D} + B q_{4D} + C \\ \text{subject to} & q_{4D}^T W q_{4D} = 1 \end{cases}, \quad (6)$$

where A, B and C depend on the matched points and

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (7)$$

To solve this problem, the Lagrangian multiplier λ can be used, resulting in the following function to minimize:

$$L(q_{4D}) = q_{4D}^T A q_{4D} + B q_{4D} + C + \lambda(q_{4D}^T W q_{4D} - 1). \quad (8)$$

The global minimum can then be found by finding the roots of a four degree polynomial in λ. Final, a 3D pose can be computed using

$${}^e\hat{q}_t = [x\, y\, \theta]^T = [x\, y\, \text{atan2}(s, c)]^T. \quad (9)$$

The expression of the covariance matrix ${}^e\Sigma_t$ can be computed from the derivatives of ${}^e\Sigma_t(q_{4D})$ in function of ${}^e\hat{q}_t$.

C. Minimization using pseudo-inverse matrix

Another way to solve (2), proposed by Low [6], is to assume that the angular variation between two consecutive iterations of the ICP is small. Therefore, we can approximate $\cos\theta \approx 1$ and $\sin\theta \approx \theta$. Using this approximation the problem becomes a linear least-squares problem:

$$\min_q E(q) = \min_q \|Aq - b\|^2, \quad (10)$$

where A and b depend on the matched points. A pseudo-inverse matrix can be used to solve the minimization problem 10:

$${}^e\hat{q}_t = \text{pinv}(A)b. \quad (11)$$

TABLE I: *Comparison of the two minimization methods with the four matchings. The accuracy is evaluated from the mean of the norms of the position errors $\|\epsilon\|$ and the mean of the absolute value of the orientation error $|\epsilon_\theta|$.*

		ICP	ICPP	PLICP	mixICP
Polynomial Minimization	$\|\epsilon\|$ (cm)	8.2	7.8	13.7	11.0
	$\|\epsilon_\theta\|$ (°)	2.97	2.84	5.97	5.26
	consistency (%)	85.5	58.8	69.8	70.0
Pseudo-inverse Minimization	$\|\epsilon\|$ (cm)	8.2	7.8	11.5	10.8
	$\|\epsilon_\theta\|$ (°)	2.94	2.83	5.64	5.24
	consistency (%)	93.5	83.9	91.6	89.8

The covariance matrix as defined by (3) is easy to compute here ($n > 3$):

$${}^e\Sigma_t = \frac{E(\hat{q})}{n-3}\left(A^T A\right)^{-1}. \quad (12)$$

IV. Simulation results[1]

We used simulated data to test different parameters with the four matching and the two minimization algorithms. Within the reference frame of the ego-vehicle, the target vehicle is placed 10 meters ahead, i.e., ${}^eq_t = [10\,0\,0]^T$. Gaussian noise has been added to these poses with the standard deviation: $\sigma_x = \sigma_y = 0.5\,\text{m}$ and $\sigma_\theta = 5°$. The LiDAR points have also been simulated with a Gaussian noise added to the range of the LiDAR beams, $\sigma_\rho = 0.1\,\text{m}$.

To test the consistency of the estimated covariance matrix ${}^e\hat{\Sigma}_t$ associated to an estimated relative pose ${}^e\hat{q}_t$ at a given risk $\alpha = 5\%$, we check if the ratio of epochs where the following inequality holds is equal to $1 - \alpha$:

$$({}^eq_t - {}^e\hat{q}_t)^T {}^e\hat{\Sigma}_t^{-1} ({}^eq_t - {}^e\hat{q}_t) < \chi^2_{3,1-\alpha}, \quad (13)$$

where $\chi^2_{3,1-\alpha} = 7.81$ for a three dimensional problem with an error probability $\alpha = 5\%$.

A. Comparison of the two minimization algorithms

One can see on table I the average position error $\|\epsilon\|$ is similar for the two minimization methods excepted for the point-to-line matching where the pseudo-inverse approach is more accurate. The pseudo-inverse method with the point-to-line matching becomes also more consistent. ICP converges to ICPP when the model is very discretized. Its large uncertainty ellipse comes from the minimization error, which is not computed with the shortest distances to the model, but with distances to the points of the model.

B. Bounding box model

In the results of table I, we used the polygonal model, in blue in figure 3. In many works, only the bounding box, in red in figure 3, is known and used.

One can see in table II that the relative poses found are less accurate in position but more accurate in orientation. The back of the bounding box is indeed not curved like the back of the polygonal model. The LiDAR scan is more

[1]The Matlab source-code used for this paper is available at: `https://www.hds.utc.fr/~heryelwa/dokuwiki/en/start`

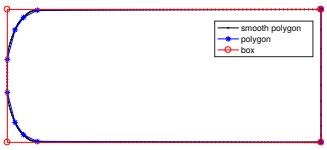


Fig. 3: *Bounding box model, in red, polygonal model, in blue, and smooth polygonal model used to simulate the LiDAR scan. As this model is used for platooning, i.e. only the back and the sides of the vehicle are in the field of view of the LiDAR, only the rear of the vehicle is detailed in the discretization of these models.*

TABLE II: *Results with a bounding box. The pose is supposed to be found when the covariance matrix can be computed without numerical singularity, when the problem has enough constraints.*

		ICP	ICPP	PLICP	mixICP
Polynomial Minimization	$\|\epsilon\|$ (cm)	12.1	9.5	9.9	9.9
	$\|\epsilon_\theta\|$ $(^\circ)$	2.36	3.39	2.97	2.97
	consistency (%)	99.9	31.1	59.9	59.9
	found (%)	100	100	56.4	56.4
Pseudo-inverse Minimization	$\|\epsilon\|$ (cm)	17.1	9.5	11.0	11.0
	$\|\epsilon_\theta\|$ $(^\circ)$	2.39	3.41	4.32	4.32
	consistency (%)	78.1	60.1	64.5	64.4
	found (%)	100	100	99.6	99.6

constrained in orientation and less constrained in position when a bounding box is used instead of a polygonal model.

When using a bounding box model, the problem is less constrained and only one segment can be matched by all the LiDAR points. In this case, the point-to-line matching is not appropriate for the first minimization. With this model and this matching, the pseudo-inverse approach found a pose with a computable covariance matrix more often.

C. Convergences of different iterative methods

Figure 4 shows an example of convergence for one epoch for the four different matching methods. When a threshold of $1cm^2$ is used, all the matching methods converge with three or four iterations on average. The ICP error is larger because it does not use the smallest distance to the model like the other methods. Sometimes, the ICPP seems to converge but after a small variation of the error during several iterations, a smaller error may still be found. When the threshold is $0cm^2$, the ICPP need 243 iterations on average to converge. Indeed, the point-to-point matching methods like the ICP or the PLICP limit the motion of the LiDAR points during the minimization of one iteration. The ICPP recomputes the points to match at every iteration, these points are increasingly closer to the final result, but the error is also increasingly smaller. When point-to-line matchings are used in the PLICP, the LiDAR points have more freedom and the convergence is faster.

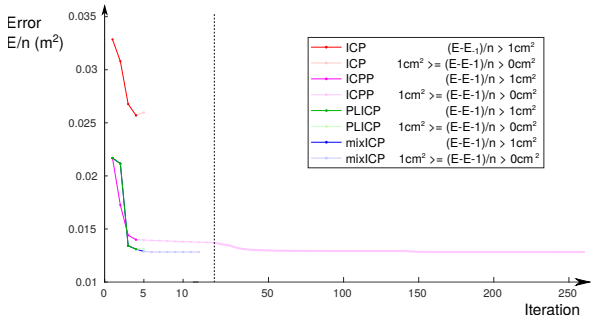
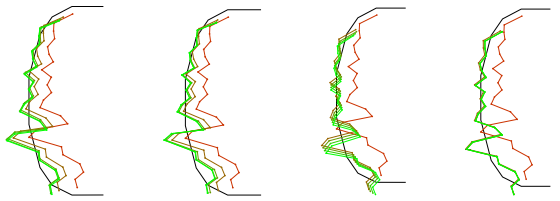


Fig. 4: *Convergences of the minimization errors for the second minimization method and the four different matchings methods. The dark color corresponds to the error before the convergence, for the $1cm^2$ threshold used to stop the algorithm. The variation of the error between two iterations divided by the number matched LiDAR points is compared to this threshold. The light color shows the error before the convergence if the threshold is $0cm^2$.*



(a) ICP (b) ICPP (c) PLICP (d) mixICP

Fig. 5: *LiDAR points convergence on the geometrical polygonal model for the different matchings. The model is shown in black and the LiDAR points change from red, before the first iteration, to green, once the algorithm has converged. 0.0001 used as threshold.*

Figure 5 shows the convergence of the LiDAR points for the four matchings. When a point-to-line matching is used in the PLICP or in the mixICP, the LiDAR points can slide along the model as shown by subfigure c.

D. Scenarios

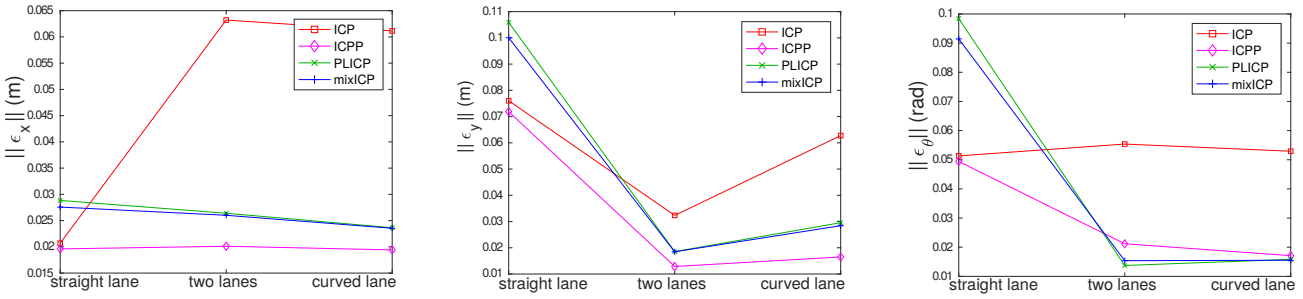
Three different scenarios have been tested on a straight road, a two lanes road and curved road. In the straight road, only the back of the vehicle is detected. The back is slightly curved, a rotation and translation invariance are present. The problem is here badly conditioned. The estimated pose is therefore not very accurate and it is more difficult to obtain the consistency. In a curved road driving, the back and one side of the vehicle are in the field of view of the LiDAR. The estimated pose is accurate. In the two lanes scenario, the back and one side of the vehicle are detected (like for the curved road) and the results are similar.

Figure 6 validates the previous hypothesis. The accuracy for y and θ increases largely when two faces of the vehicle are in the field of view of the LiDAR, e.g., in the two lanes and the curved lane scenarios.

In the figure 7, one can see that the consistency is higher for the two lanes driving than for the two other scenarios.

E. Inter-distance dependency

The previous results has been computed for an inter-distance between the leader vehicle and the follower of 10m. We have also tested the dependency of the error and the consistency for different inter-distances. When the inter-distance increases, the number of LiDAR points on the leader decreases. The error and the consistency become larger (Fig. 8).



(a) $|\bar{\epsilon_x}|$ (b) $|\bar{\epsilon_y}|$ (c) $|\bar{\epsilon_\theta}|$

Fig. 6: *Mean of the absolute value of the errors for x, y and θ for the four matchings and for the straight lane, the two lanes and the curved lane drivings.*

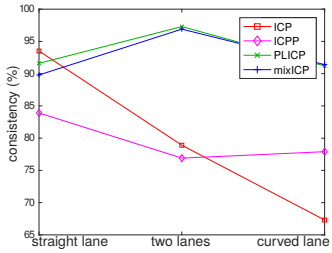
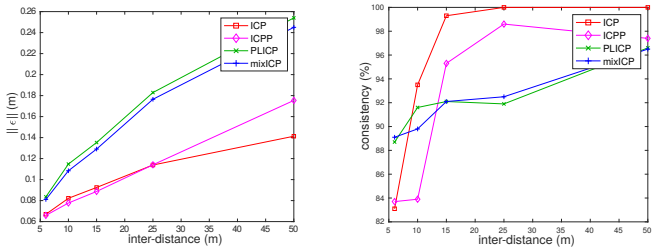


Fig. 7: *Consistency for the four matchings and for the three scenarios: the straight lane, the two lanes and the curved lane drivings.*



(a) *Norm of the position error* $\|\bar{\epsilon}\|$. (b) *Consistency.*

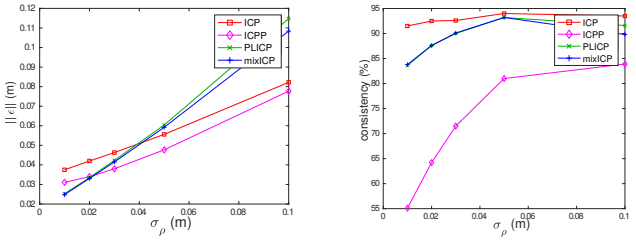
Fig. 8: *Mean of the norm of the relative position error $\|\epsilon\|$ and consistency for the four matchings depending on the inter-distance.*

F. LiDAR noise dependency

The previous results are tested with a range noise on the LiDAR points with a standard deviation of 10cm. We test in this section the error and the consistency when the LiDAR becomes more accurate. The error and the consistency increase when the LiDAR noise increases. (Fig. 9).

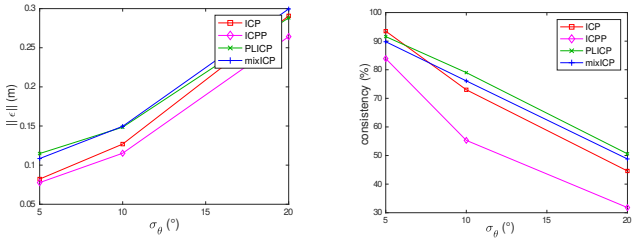
G. Noise on the poses of the leader and the follower

The standard deviations $[\sigma_x\, \sigma_y\, \sigma_\theta] = [0.5m\, 0.5m\, 5°]$ are applied on the poses of the follower and the leader vehicles on the previous results. We test here the effect of less accurate poses on the error and the consistency. Even if the initial localization corrects greatly the position error, the iterative minimization is very sensitive to the orientation



(a) $\|\bar{\epsilon}\|$ (b) *Consistency.*

Fig. 9: *Mean of the norm of the relative position error $\|\epsilon\|$ and consistency for the four matchings depending on the LiDAR range noise with a standard deviation σ_ρ.*



(a) $\|\bar{\epsilon}\|$. (b) *Consistency.*

Fig. 10: *Mean of the norm of the relative $\|\epsilon\|$ and consistency for the four matchings depending on the noise apply to the poses of the follower and the leader vehicles with the standard deviations:* $[\sigma_x\, \sigma_y\, \sigma_\theta] = [0.5\, 0.5\, 5]$, $[\sigma_x\, \sigma_y\, \sigma_\theta] = [2.5\, 2.5\, 10]$ *and* $[\sigma_x\, \sigma_y\, \sigma_\theta] = [5\, 5\, 20]$ *($[\sigma_x] = [\sigma_y] = m$ and $[\sigma_\theta] = °$).*

noise. When it increases, the error increases (Fig. 10a) and the consistency decreases drastically (Fig. 10b). If the relative orientation error becomes very large (near 45°) some ambiguity can appear and the LiDAR points can match on the wrong side of the vehicle.

In this simulation, only one other vehicle was present. If two or more vehicles are present the position error can creates some ambiguity when the algorithm has to choose which points matches with which vehicles.

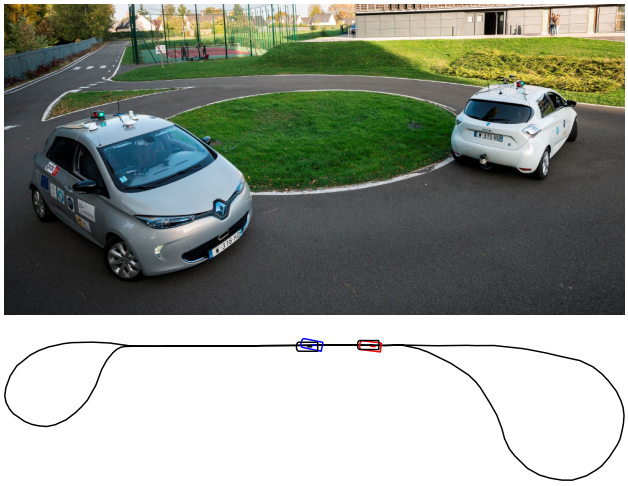


Fig. 11: *Real platooning scenario on the test track Seville, Compiègne, France, in black. The ground truth of the two vehicles are shown in black with their models. The pose and the model are in red for the leader and in blue for the follower.*

TABLE III: *Real platooning scenario errors $\|\epsilon\|$ and consistency for the four matchings, for second minimization method.*

	ICP	ICPP	PLICP	mixICP
$\|\epsilon\|$ (cm)	8.5	5.3	7.1	6.8
consistency (%)	75.1	68.6	92.5	91.8

H. Real platooning scenario

The relative localization have been tested on a real platooning scenario. Two vehicles of the laboratory were driving together on a test track (Fig. 11). This track has two roundabouts and one straight lane between them.

Both vehicles were equipped with an IMU (Span CPT) with a GNSS receiver using RTK corrections for the ground truth. In practice, we have noticed that this ground truth was not accurate enough for relative localization compared to the high quality of the LiDAR measurements. Therefore, the LiDAR of the follower was not used but simulated to correspond perfectly with the pose given by the GNSS receiver. The poses of the follower and of the leader used for the relative localization algorithms were the ground truth with Gaussian noise such as : $[\sigma_x\, \sigma_y\, \sigma_\theta] = [0.5m\, 0.5m\, 5°]$. The LiDAR was simulated with a Gaussian range noise with a 10cm standard deviation.

In this scenario, the inter-distance was evolving between 6m and 16m. The vehicles were following each other on the curved road of the roundabout and on the straight lane.

The consistency and the accuracy are similar to the other results : 92.5% of consistency and 71 mm of error for the point-to-line matching.

V. Conclusion

This work has presented different relative localization methods based on LiDAR points. An estimated pose received from the detected vehicle is used for initialization. A first localization is computed using the bounding boxes of the LiDAR points and of the communicated model. This is used to reduce the position error, the orientation error being unchanged in this stage. An iterative minimization algorithm is then applied using this first localization. We have presented and compared two minimization methods and four different points to polygonal model matchings. First, we have noticed on different scenarios that the second minimization method using the pseudo-inverse matrix gives a better accuracy and consistency. Secondly, a point-to-line matching allows a better estimation of the covariance matrix. This matching gives more freedom to the LiDAR points which can slide along the model. Moreover, this matching needs less iterations to converge. We have also observed that, when two sides of a vehicle are in the field of view of the LiDAR, the problem is better conditioned and the accuracy is higher.

In future work, we will use these algorithms to compute the absolute pose of the follower using the estimated pose of the leader like a deported GNSS antenna. We will test these algorithms with experimental data with two vehicles and more.

Acknowledgment

This work was carried out in the framework of Equipex ROBOTEX (ANR-10- EQPX-44-01) and Labex MS2T (ANR-11-IDEX-0004-02). It was also carried out within SIVALab, a shared laboratory between Renault, CNRS and UTC.

References

[1] O. Bengtsson and A.J. Baerveldt. Robot localization based on scan-matching-estimating the covariance matrix for the IDC algorithm. *Robotics and Autonomous Systems*, 44(1):29–40, July 2003.

[2] A. Censi. An accurate closed-form estimate of ICP's covariance. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3167–3172, April 2007.

[3] A. Censi. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*, pages 19–25, May 2008.

[4] T. Chateau, F. Chabot, C. Teulière, M. Chaouch, and J. Rabarisoa. Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2d and 3d Vehicle Analysis from Monocular Image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, United States, 2017.

[5] ETSI. Intelligent transport systems (ITS); Vehicular communications; Basic set of applications; Part 2: specification of cooperative awareness basic service. Technical Report ETSI EN 302 637-2 v1.3.1, Sep. 2014.

[6] K.L. Low. Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. Technical Report TR04-004, Department of Computer Science University of North Carolina at Chapel Hill, February 2004.

[7] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, April 2009.

[8] S. M. Prakhya, L. Bingbing, Y. Rui, and W. Lin. A closed-form estimate of 3d ICP covariance. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pages 526–529, May 2015.

[9] A. Shetty. *GPS-LiDAR Sensor Fusion Aided by 3D City Models for UAVs*. M. S. Thesis, University of Illinois, Urbana-Champaign, 2017.

[10] X. Zhang, W. Xu, C. Dong, and J. M. Dolan. Efficient L-shape fitting for vehicle detection using laser scanners. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 54–59, June 2017.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session IV

Interactive session

- **Title: Vehicle Detection in UAV Aerial Video**
  Authors: H. Zhang, C. Meng, P. Guo, X. Ding and Z. Li
- **Title: MOMDP solving algorithms comparison for safe path planning problems in urban environments**
  Authors: J.A. Delamer and Y. Watanabe and C. P. Carvalho Chanel
- **Title: On finding low complexity heuristics for path planning in safety relevant applications**
  Authors: R. Krutsch
- **Title: Enhancing the educational process related to autonomous driving**
  Authors: N. Sarantinoudis, P. Spanoudakis, L. Doitsidis and N. Tsourveloudis
- **Title: CoMapping: Efficient 3D-Map Sharing: Methodology for Decentralized cases**
  Authors: L. F. Contreras-Samame, S. Dominguez-Quijada, O. Kermorgant and P. Martinet
- **Title: Single-View Place Recognition under Seasonal Changes**
  Authors: D. Olid, J. M. Facil and J. Civera
- **Title: Future Depth as Value Signal for Learning Collision Avoidance**
  Authors: K. Kelchtermans and T. Tuytelaars
- **Title: Automatic generation of ground truth for the evaluation of obstacle detection and tracking techniques**
  Authors: H. Hajri, E. Doucet, M. Revilloud, L. Halit, B. Lusetti, M.C. Rahal
- **Title: Autonomous navigation using visual sparse map**
  Authors: S. Hong and H. Cheng
- **Title: Socially Invisible Navigation for Intelligent Vehicles**
  Authors: A. Bera, T. Randhavane, E. Kubin, A. Wang, K. Gray, and D. Manocha
- **Title: An Egocubemap Based Algorithm for Quadrotors Obstacle Avoidance Using a Single Depth Camera**
  Authors: T. Tezenas Du Montcel, A. Negre , M. Muschinowski , E. Gomez-Balderas and N. Marchand
- **Title: DisNet: A novel method for distance estimation from monocular camera**
  Authors: M. Abdul Haseeb, J. Guan, D. Ristić-Durrant, A. Gräse

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Vehicle Detection in UAV Aerial Video[a]

Huan Zhang[1], Cai Meng[1*], Pin Guo[2], Xilun Ding[2] and Zhaoxi Li[1]

***Abstract*—In recent years, unmanned aerial vehicle (UAV) has been increasingly applied to traffic monitoring field, especially the vehicle detection. However, there are many challenges for vehicle detection in the UAV aerial video, such as camera shake, interferential targets and a wide range of change of scene. To solve these problems, a new vehicle detection system which is suitable for detection in UAV aerial video is proposed. We use the bit plane to extract the lane surface and use the extracted lane surface to limit the detection area. We improve the Vibe algorithm so that it can be used in rapidly changing scenes. In addition, the moving target screening strategy is proposed to screen moving vehicles. This paper is the first one to introduce bit plane into detection method. Our novel detection system is another major contribution. Experiments show that our system outperforms existing detection algorithms in terms of accuracy and computation time.**

I. INTRODUCTION

In recent years, with the prosperity of the transportation industry, lane detection, vehicle detection and vehicle classification, etc. have become the most popular areas in the field of traffic monitoring [1]. At the same time, unmanned aerial vehicles (UAV), as a platform for collecting data, have been increasingly applied to traffic monitoring and other fields.

The application of UAV in vehicle detection has a series of advantages, such as flexibility, friendliness, adjustable traffic monitoring range, and on-demand image acquisition. However, there are also a series of challenges: (1) During flight, UAV could be affected by the weather. The UAV will shake and drift, so the aerial video will be swaying and distorting. (2) Due to the wide shooting range of UAV, aerial video will inevitably be introduced into the large background area, which will interfere with vehicle detection [2]. (3) UAV moves quickly and flexibly, resulting in rapid changes in background. (4) There are pedestrians and other moving interference targets. Because of these problems, the detection in aerial video needs to be robust, anti-interference and real-time.

In dynamic scenarios, motion based methods, such as the optical flow method and the motion vector based method, are expensive in computing and difficult to detect real-time. In the feature based method, the simple feature method is less robust, and the subarea method is complex, which is not suitable for real time detection. Deep learning based method, such as HDNN [3] and CNN system combined with SVM classifier [4], are used to detect in UAV images rather than aerial video. The background modeling method has faster detection speed than the feature based and motion based methods, and detect the moving target accurately.

In this paper, we study the best real-time algorithm in the current detection algorithm [20], the Vibe algorithm [14]. Many researchers have studied and optimized the algorithm. Li, et al. [5] integrated the adjacent frame difference with Vibe algorithm to remove the ghost area. The PBAS algorithm combined the advantages of the SACON algorithm and the Vibe algorithm, and optimized on these basis [6]. Ehsan Mahoor, et al. changed the pixel value in the sample set to the frame difference value, so even if the camera is jitter, their algorithm is still effective [7]. Jin and his colleges fused the improved Canny operator with the Vibe algorithm to get a more accurate foreground region [8].

Unfortunately, most of the improved Vibe algorithms are still limited to detect targets in static scenes, which is not suitable for detecting targets in moving aerial video. As mentioned, due to the large range of aerial video, the moving background area can be easily detected as the foreground target. In order to reduce the background complexity, we used lane information to restrict the detection area dynamically. Because of dramatic changes in the scene, we improved the Vibe algorithm to enable it to detect in a rapidly changing environment. There are interference targets, such as pedestrians. We proposed a series of vehicle screening criteria to screen these targets. To the author's knowledge, this paper is the first one to introduce bit plane into detection method. In addition, the design of our vehicle detection system, which is suitable to detect in aerial video, is the paper's main contribution as well.

II. DYNAMIC DETECTION AREA

The scene in UAV aerial video often changes, and the detection area should be changing accordingly. In order to reduce the background complexity and limit the shooting range, the detection area should be limited to the lane surface. The traditional fixed ROI areas are not suitable for aerial video for they are updated very slowly or not updated. Therefore, we propose our dynamic detection area using the lane information.

The current lane detection methods include feature-based methods and model-based methods. In the model-based methods, the mathematical models of lane lines are established and optimized, such as line model, two degree curve model and hyperbolic model, etc.[9].Feature-based methods use the color, texture, margin, etc. to extract lane. Most lane detection algorithms are designed according to the specific system tasks and application environments [10] [11] [12]. These algorithms are not universal when it comes to harsh environments such as raining or nightscape. Therefore, we design a lane detection

a: Thanks to the support by NFSC with Grants no. 91748201, and by the sponsorship from the Mohamed Bin Zayed International Robotics Challenge 2017

1: School of Astronautics, Beihang University, Beijing, China.

2: School of Mechanical and Automation, Beihang University, Beijing, China

*Corresponding author: tsai@buaa.edu.cn

algorithm based on bit plane, which is composed of lane surface extraction, lane detection and lane tracking.

A. *Lane Surface Extraction*

For further lane detection, frames are usually thresholding to generate the binary images, and edges are extracted by Canny operator on the binary images. In the thresholding, the integrity of the lane surface is damaged, which will affect further lane detection. Therefore, we introduce bit planes to maintain road integrity. The bit plane describes the grayscale attribute of the image. The grayscale of a m bit image can be expressed as a polynomial:

$$P_m = a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_0 2^0 \qquad (1)$$

$a_i (i = 0,1, \ldots m-1)$ is the coefficient of the bit planes. An 8 bits image can be considered as 8 planes of 1 bit, including 4 high-older planes and 4 low-older planes. The high-older planes contains important information, such as the contours. The low-older contains the details of the image, as shown in Figure 1. In aerial video, the details of images change greatly, which is considered to be interference in dynamic area detection. In order to avoid small grayscale changes influencing the lane surface extraction greatly, we only use high old bit plane to get the dynamic area.

In different application scenarios, we choose bit planes according to illumination conditions. The 8th bit plane is usually used to extract the road surface. In raining or night scape, we use the 7th bit plane. Even in bad weather and night scenes, the lane surface extraction results of bit plane are still better than the Ostu results. The experimental section will introduce the details of the experiment.

On the selected bit plane, we calculate the vanishing point of the road, i.e. where the lane vanishes. By calculating the average grayscale μ_i of row i in the image (m rows and n columns), we get the vanishing point of lane.

$$\mu_i = \frac{\sum_{j=0}^{n} v_{i,j}}{n_i}, (i = 1,2, \ldots m, j = 1,2, \ldots n) \qquad (2)$$

$v_{i,j}$ represents the pixel grayscale. On the vanishing point, the average grayscale μ_i decrease sharply:

$$|\mu_{i-1} - \mu_i| > T_1 \qquad (3)$$

The threshold T_1 is set according to the specific application scenario. The pixels above the horizon are undesired and these pixels will be set to zero.

B. Lane Detection

After extracting the lane surface, the Canny operator is used to detect edges on the binary 8th bit plane images. After the probability Hough transform, the straight line can be expressed as: $\rho = x \cdot cos\theta + y \cdot sin\theta$. ρ is the distance from the line to the origin of the coordinate system. θ is the angle between the line and the x axis.

$$l = \begin{cases} l_f & \theta \epsilon (90°, 180°) \\ l_r & \theta \epsilon (0°, 90°) \end{cases} \quad if\ \theta < 0°, \theta = \theta + 180° \qquad (4)$$

According to the angle θ, the straight line would be determined whether it belongs to the left lane l_f or the right lane l_r, and the two lateral lines are collected respectively. Angle information is used to screen out the outermost lane on both sides. Finally, the outmost point P_L, P_R of the outermost lanes of the left and right sides are used as the rectangle corners to achieve the dynamic detection area. The use of rectangles instead of other curves is to facilitate lane tracking and limit computation complexity.

C. Lane Tracking

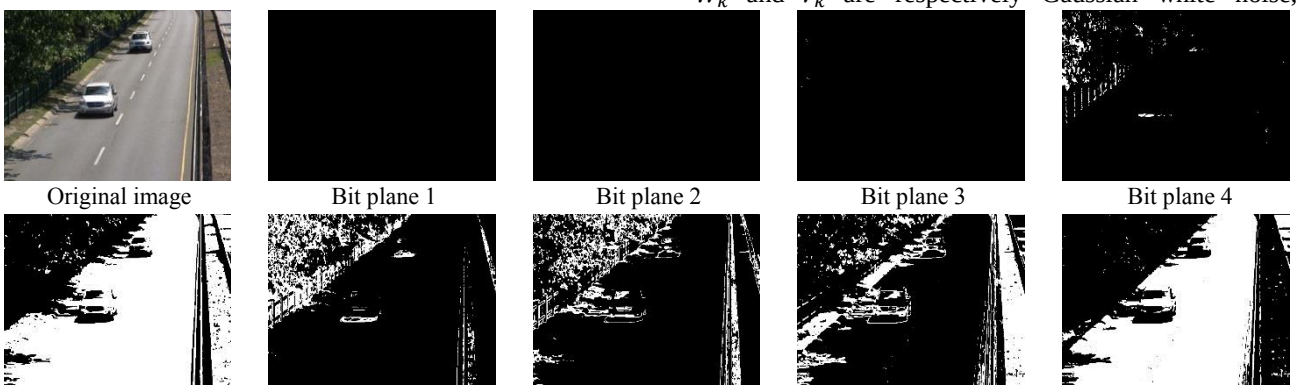
Due to the jitter and mutation in aerial video, the dynamic detection areas between adjacent frames will change greatly. Therefore, the object located at the edge of the lane can be easily detected as a vehicle. Here, we use Kalman filtering to limit area variance [13]. Traditionally, Kalman filter is used to track ρ and θ of lanes [21]. In our method, we track the outermost point of the lane. Take the left lane for example. Kalman filter uses previous state and current measurement to predict the optimal state. Lane tracking system can be described by linear stochastic differential equation:

$$X_k = AX_{k-1} + BU_k + W_k \quad Z_k = HX_k + V_k \qquad (5)$$

The position and its variants are defined as the state vector X, and the position is defined as the measurement vector Z:

$$\mathrm{X} = [X_{Pl}\ Y_{Pl}\ X_{Pl}'\ Y_{Pl}']^T \quad \mathrm{Z} = [X_{Pl}\ Y_{Pl}]^T \qquad (6)$$

This 4×4 state updating matrix A is set to the unit matrix. The measurement matrix H is set to the identity matrix. Both W_k and V_k are respectively Gaussian white noise, i.e.



Original image | Bit plane 1 | Bit plane 2 | Bit plane 3 | Bit plane 4

Otsu threshold | Bit plane 5 | Bit plane 6 | Bit plane 7 | Bit plane 8

Figure 1. The binarization results for lane surface extraction. The top left is the original image. The bottom left is the binary image by the Otsu threshold. The others are the binary images by bit planes, ranging from bit plane 1 to bit plane 8. The high-older bit planes contains the main information of lane and the low-older bit planes contains the details.

$W_k \sim N(0, Q)$, $V_k \sim N(0, R)$. The Kalman filter is used to track the outmost points of lanes as follows.

The current state is predicted by using the optimization result of the former state $X_{k-1|k-1}$. There is no control, so the U_k is set to 0. Q represents the noise matrix of the system, which is set to the identity matrix in the application. The measurement noise matrix R is set as an identity matrix. The specific calculation steps are as follows:

$$X_{k|k-1} = AX_{k-1|k-1} + W_k \tag{7}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{8}$$

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - HX_{k|k-1}) \tag{9}$$

$$K_k = P_{k|k-1}H^T / (HP_{k|k-1}H^T + R) \tag{10}$$

$$P_{k|k} = (I - K_k H)\, P_{k|k-1} \tag{11}$$

Through these steps, lane tracking is completed by the predicting of the outmost points of the lanes. Since each Kalman prediction requires only one iteration, the time cost is negligible. By designing the dynamic detection area, the background is effectively eliminated. The complexity of vehicle detection is reduced.

III. Improved Vibe Algorithm

To the best of our knowledge, Vibe algorithm is used for the video surveillance in static background. When applied to the detection of dynamic scenes, there will be slow background updates and ghost problems [14] [15]. Therefore, we improved the Vibe algorithm and applied it to the dynamic scenes.

In Vibe, only one frame is needed to build the background model. By using adjacent pixels with spatial consistency, N samples are randomly selected from the eight neighborhoods as their sample set. The sample set of the whole image pixel is the initial background model. The sample set of any pixel $M(x, y, t)$ in the image is $M(x, y, t) = \{v_1, v_2, \dots, v_i, \dots v_N\}$, $v_i (i = 1,2, \dots N)$ is the pixel value of the sample set point.

The pixel $M_1(x, y, t)$ in the current frame is defined as the center of a circle, and the R is defined as the radius of the circle recorded as $SR(M_1)$. Compare $M_1(x, y, t)$ with the pixel of the same position in the previous frame M. If the intersection point is greater than the threshold T_{Vibe}: $\#\{SR(M_1) \cap M\} > T_{Vibe}$, the current point $M_1(x, y, t)$ is the background point, and enter the background update step.

When the current pixel is determined as a background pixel, it has a probability of $1/\varphi$ to update itself (φ is usually set to16), i.e., a pixel value v_i is randomly selected from the N samples to be updated. It also has a probability of $1/\varphi$ to update the neighbor pixels. The probability of sample point remained in the sample set decreases exponential over time. The background model can be updated quickly and effectively. When a pixel is continuously determined as foreground pixel of multi frames, it is updated to background pixel. That is to prevent the phenomenon of deadlock.

A. Improvement

In static environment, the background is almost unchanged, and the Vibe algorithm is robust. However, when the UAV moves quickly, the camera jittering, the background changes drastically, and the amount of the foreground increases sharply. The ghost appears, and the false objects are eliminated very slowly. We use morphological open operation to eliminate the noise and then we use morphological close operation to fill the holes. At the same time, the previous scene model is not suitable for the current scene, so we set different update rates of $1/\varphi$ based on the complexity of the scene to speed up the elimination of false targets. In the current frame, the number of foreground targets is defined as $\sum_t^i$:

$$1/\varphi = \begin{cases} 1/16, & \sum_t^i \le 10 \\ 1/4, & 10 < \sum_t^i < 20 \\ 1/2, & \sum_t^i \ge 20 \end{cases} \tag{12}$$

As shown in Table 2, the scene updates faster with the increase of $1/\varphi$.

IV. Vehicle Screening Method

After detection in dynamic detection area, a series of moving targets will appear. Therefore, we filter the moving targets to get vehicles. In aerial video, the widely used vehicle screening features, such as color, texture, shadow, texture, symmetry, are easily affected by illumination and other factors. Therefore, we provide our screening criteria: scale criteria, shape criteria, and direction criteria.

A. Scale Criteria

Vehicle detection is usually carried out at a fixed scale, and the scale characteristics of vehicles are generally ignored. The size of moving target in aerial video is related to the flight height of UAV. When the flight altitude is low, the moving target will be correspondingly large, and the smaller foreground target is the unwanted object, and vice versa. Therefore, UAV flight height can be used to determine the size of vehicle. Let $V(x, y, t)$ represent the current moving target area. V_L and V_H are low threshold and high threshold of the vehicle area. H is defined as the flight height. V_L and V_H are the functions of H. We use pinhole camera models to build the projection models.

The corresponding relation between the point coordinates in the real world coordinate system $P(x, y, z)$ and the image pixel coordinates system (u, v, d) (d is depth data) can be described as:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = C \cdot \left(R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \right) \quad d = z \cdot s \tag{13}$$

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

R and T are the camera attitudes. R represents the rotation matrix and T represents the translation matrix. C represents the internal parameter matrix of the camera. s refers to the scale factor of depth. f_x, f_y represent the camera focal length on the x and y axis in the camera coordinate system. c_x, c_y refer to the camera aperture center. f_x, f_y, c_x, c_y could be

obtained by camera calibration. Since the pod can be self-stabilized, we set the pod to always look down, and the optical axis of the pod is perpendicular to the ground. Therefore, the camera does not rotate and translate. R is set to a unit matrix I, and T is set to zero. The above formula can be simplified to:

$$u = x \cdot f_x/z + c_x \quad v = y \cdot f_y/z + c_y \tag{15}$$

We use the laser on the UAV to obtain the distance z between the pod and the ground, which is similar to the flight height H. Utilizing actual vehicle width and height, the vehicle size threshold V_L and V_H in the image can be obtained. If the size of current target is:

$$V \ll V_L \text{ or} V \gg V_H \quad V_L = f(H), V_{H=}f(H) \tag{16}$$

It wouldn't be the candidate vehicle targets. Remove current targets. The threshold V_L and V_H are set according to the specific application scenarios.

B. Shape Criteria

It is generally believed that the shape of the target vehicle is rectangular. Therefore, the ratio of target length l_t^i and target width w_t^i can be used to filter targets:

$$r = l_t^i/w_t^i \tag{17}$$

When $r > 6$ or $r < 0.5$, it is not a candidate vehicle target.

C. Direction Criteria

In the lane where there is no crossing, the vehicles don't move back and forth, so we propose the direction criteria: the motion direction of the candidate vehicle is monotonous. We introduce cross products to determine the direction. The algorithm is illustrated as follows:

Algorithm The direction criteria

1: Take the target t of frame i as $P_t^i(x_t^i, y_t^i)$ ($i = 1,2,3 \ldots \ldots N-2$, N is the amount of frames);

2: Take the same moving target t of adjacent frames as $P_t^{i+1}(x_t^{i+1}, y_t^{i+1})$ and $P_t^{i+2}(x_t^{i+2}, y_t^{i+2})$ calculate the cross product: $cp_i = |\overrightarrow{P_t^i P_t^{i+1}} \times \overrightarrow{P_t^{i+1} P_t^{i+2}}|$ and record the result;

3: If cp_i is different from the sign of posterior cross product cp_{i+1}, the direction of movement are different. The target isn't considered as vehicle.

4: Repeat for the frames.

V. EXPERIMENT

We carried out several experiments to evaluate our vehicle detection system on aerial video and changedetection.net (CDnet) [21]. CDnet is one of the most widely used dataset for moving object detection. Experiment results are as follows.

A. UAV Platform

The UAV is designed and assembled independently by us. We use the DJI S900 UAV frame and 16000mAh, 6S LiPo battery. Flight control system consists of the STM32F427VI and TLC algorithm [16] [17] [18] [19]. The IMU unit uses the Xsens MTI-G-700. IMU and GPS collect data and calculate the necessary speed, position, angular velocity and attitude of UAV, so as to provide necessary data for flight control. The XBee module realizes real-time data exchange between ground station and UAV. The distance between UAV and moving target is obtained by using SF11 laser altimeter. The image acquisition device is a double axle pod, and the image processing algorithm is completed on the airborne PC104. The overall UAV system and parameters are shown in Figure 2 and Table 1.

Table 1. Hardware system parameters.

Frame Weight	Diagonal Wheelbase	Max Distance of Propeller
3300g	900mm	1200mm
Load	Height	Wind Resistance
2000g	480mm	F-5
Working Temperature		Max Speed
-10 °C ~ +40 °C		10m/s
Max Flight Time		
25min（16000mAh/6s/1900g）		

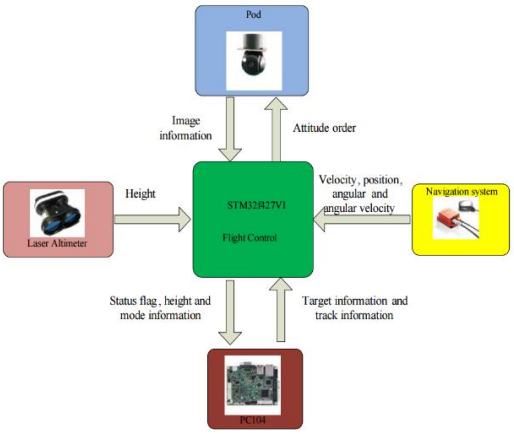


Figure 2. UAV physical map and UAV hardware diagram.

B. Lane Surface Extraction Experiment

As shown in Figure 1, the result of lane surface extraction based on bit plane is superior to the Ostu binarization result on the highway dataset (1700 frames). In order to verify the performance of the lane surface extraction method based on the bit plane in different environments, we test on traffic datasets (1570 frames) and fluidHighway dataset (1364 frames). Even if the scene of these datasets is night and the image quality is terrible, the bit plane can still extract the relatively complete lane surface.

C. Dynamic Detection Area Experiment

The dynamic detection area results are shown in Figure 3. Aerial video was taken in 5m, 10m, and 20m. The black line represents the outmost lanes. The blue rectangle is the detection area obtained from the outmost points, and the purple rectangle is the final detection area after Kalman filtering. It can be seen that the detection area can be effectively reduced.

D. Improved Vibe Algorithm Experiment

The results of the improved Vibe algorithm are shown in Table 2. With the change of background pixel update rate, background removal becomes faster and more effective. The improved algorithm can be applied to dynamic environment.

Figure 3: The black straight lines are detected by the probability Hough transform and selected as the outermost lanes. The blue box is the dynamic detection area fitted by the outermost lanes. The purple box is the filtered dynamic detection area obtained by Kalman filter.

Table 2. Frame number for updating with different φ value.

Update rate $1/\varphi$	1/16	1/8	1/4	1/2
Required frame amount for updating	100	70	60	50

E. Aerial Video Experiment

We compare our algorithm with Vibe algorithm, GMM algorithm and PBAS algorithm, because the Vibe is the basis of our algorithm, the GMM is frequently cited and utilized. The PBAS is nearly the best detection algorithms in recent years [20]. We add simple morphological operation to the original Vibe and GMM to eliminate noise. The experiment results are shown in Figure 4.

The images in Figure 4 are the Vibe results, GMM results, PBAS results and our system results. The average time cost per frame for the four algorithm is 23 milliseconds, 40 milliseconds, 42 milliseconds and 30 milliseconds. Our detection system is superior to other algorithms. After combining the dynamic detection area, the improved Vibe algorithm and the vehicle screening strategy, the vehicle detection system can detect the moving vehicles accurately in the dynamic large-scale environment.

F. CDnet Experiment

We test our method on the twoPositionPTZCam dataset, which belongs to the PTZ category in CDnet. As proposed in [21], detection algorithms have the lowest performance on the PTZ category due to the camera jitter, and the camera jitter is inevitable in aerial video. The main advantage of this comparison is that we can easily compare our method with many most advanced methods which have been ranked according to the following measures. The evaluation used by the CDnet is in Table 3 and the results are in Table 4.

Table 3. Evaluation.

TP (True Positive)	Pixel number labeled as foreground correctly.
FP (False Positive)	Pixel number labeled as foreground incorrectly.
FN (False Negative)	Pixel number labeled as background incorrectly.
TN (True Negative)	Pixel number labeled as background correctly.
Re (Recall)	**TP / (TP + FN)**
Sp (Specificity)	**TN / (TN + FP)**
FPR	**FP / (FP + TN)**
FNR	**FN / (TP + FN)**
Precision	**TP / (TP + FP)**
PWC	**100 * (FN + FP) / (TP + FN + FP + TN)**
F-Measure	**(2 * Precision * Re) / (Precision + Re)**
Average ranking (used to rank by CDnet)	**(Re + Sp + FPR + FNR +PWC + F-Measure + Precision) / 7**

VI. CONCLUSION

In this paper, an UAV airborne vehicle detection system is proposed. We use the bit plane to limit dynamic detection area, reduce the background complexity and improve detection efficiency. Then the Vibe algorithm is improved so that it can be applied to dynamic scenes. Finally, the foreground targets are filtered to get the moving vehicles. Experiments show that the vehicle detection system is accurate, robust and real-time.

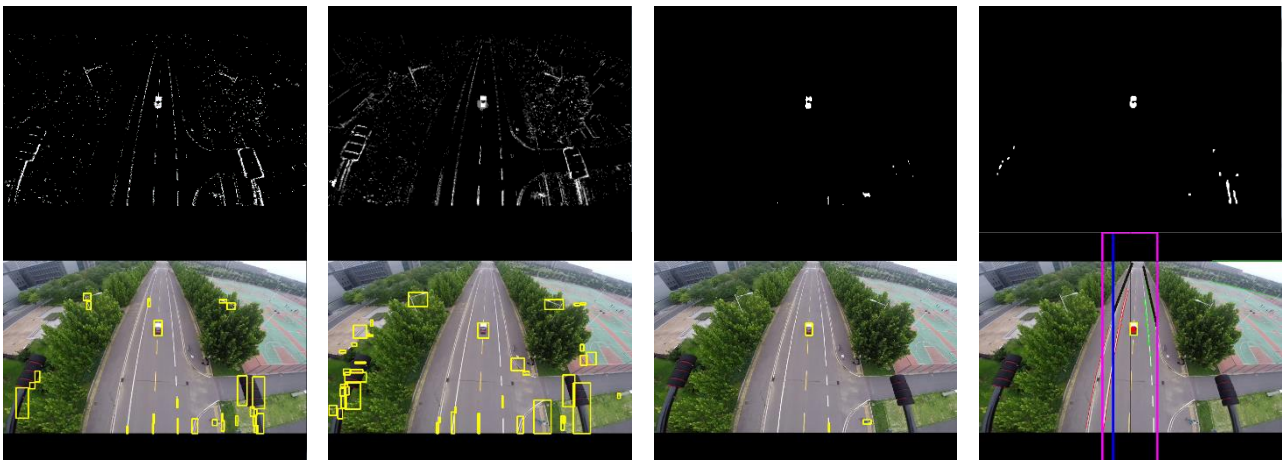


Figure 4. The images from left to right are the detection result of the Vibe algorithm, the GMM algorithm result, the PBAS algorithm result and our system result.

Table 3. Test results on the twoPositionPTZCam dataset

Method	Re	Sp	FPR	FNR	PWC	F-Measure	Average ranking
C-EFIC [22]	0.8686	0.8947	0.1053	0.1314	10.5973	0.6207	0.6144
Our method	0.6353	0.9929	0.0071	0.3647	1.1176	0.5666	0.5993
Multimode Background Subtraction [23]	0.5973	0.9963	0.0037	0.4027	0.5850	0.5520	0.5400
EFIC [24]	0.9177	0.9217	0.0783	0.0823	7.8707	0.5842	0.5282
Multimode Background Subtraction V0 (MBS V0) [25]	0.5770	0.9945	0.0055	0.4230	0.7821	0.5118	0.4988
PAWCS [26]	0.6976	0.9912	0.0088	0.3024	1.1162	0.4615	0.4725
IUTIS-5 [27]	0.6749	0.9902	0.0098	0.3251	1.2166	0.4282	0.3833
AAPSA [28]	0.5128	0.9638	0.0362	0.4872	3.9676	0.3302	0.3772

Therefore, the system has a broad application prospect. Because of the outdoor working environment, improving the robustness of lighting is a good direction for future work.

REFERENCES

[1] Hadi R A, Sulong G, George L E. Vehicle detection and tracking techniques: a concise review [J].Signal & Image Processing, 2014, 5(1):1-12.

[2] Siam, M, R. Elsayed, and M. Elhelw. On-board multiple target detection and tracking on camera-equipped aerial vehicles. IEEE International Conference on Robotics and Biomimetics, 2012:2399-2405.

[3] Chen, X, Xiang S, Liu C L, et al. Vehicle detection in satellite images by hybrid deep convolutional neural networks [J]. IEEE Geoscience and remote sensing letters, 2014, 11(10): 1797-1801.

[4] Ammour N, Alhichri H, Bazi Y, et al. Deep learning approach for car detection in UAV imagery [J]. Remote Sensing, 2017, 9(4): 312.

[5] Li Y, Chen W, Jiang R. The integration adjacent frame difference of improved ViBe for foreground object detection[C]//2011 7th International Conference on Wireless Communications, Networking and Mobile Computing. 2011.

[6] Hofmann M, Tiefenbacher P, Rigoll G. Background segmentation with feedback: The pixel-based adaptive segmenter[C]//IEEE Computer Vision and Pattern Recognition Workshops (CVPRW). 2012: 38-43.

[7] Mahoor E, Maghsoumi H, Asemani D. An improved motion detection algorithm using ViBe[C]//IEEE International Conference on Computing, Communication and Networking Technologies (ICCCNT). 2015: 1-5.

[8] Jin D, Zhu S, Sun X, et al. Fusing Canny operator with vibe algorithm for target detection[C]// IEEE Control and Decision Conference (CCDC), 2016 Chinese. 2016: 119-123.

[9] Xining Y, Jianmin D, Dezhi G, et al. Research on Lane Detection Based on Improved Hough Transform [J]. Computer Measurement & Control, 2010, 2: 019.

[10] McCall J C, Trivedi M M. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation [J]. IEEE transactions on intelligent transportation systems, 2006, 7(1): 20-37.

[11] Mammeri A, Boukerche A, Tang Z. A real-time lane marking localization, tracking and communication system [J]. Computer Communications, 2016, 73: 132-143.

[12] Rathinam S, Kim Z W, Sengupta R. Vision-based monitoring of locally linear structures using an unmanned aerial vehicle[J]. Journal of Infrastructure Systems, 2008, 14(1): 52-63.

[13] Mammeri A, Boukerche A, Lu G. Lane detection and tracking system based on the MSER algorithm, hough transform and kalman filter[C]//Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems. ACM, 2014: 259-266.

[14] Barnich O, Van Droogenbroeck M. ViBe: A universal background subtraction algorithm for video sequences [J]. IEEE Transactions on Image processing, 2011, 20(6): 1709-1724.

[15] Sun, S, Qin Y, Xianbing M A. ViBe foreground detection algorithm and its improvement with morphology post-processing for outdoor scene [J]. Computer Engineering & Applications, 2013, 49(10):159-162.

[16] Li Z, Meng C, Zhou F, et al. Fast Vision-based Autonomous Detection of Moving Cooperative Target for UAV Landing. Journal of Field Robotics, 2018, DOI: 10.1002/rob.21815.

[17] Ding X, Guo P, Xu K, et al. A Review of Aerial Manipulation of Small-scale Rotorcraft Unmanned Robotic Systems [J]. Chinese Journal of Aeronautics, 2018.

[18] Ding X, Yu Y. Motion planning and stabilization control of a multipropeller multifunction aerial robot [J]. IEEE/ASME Transactions on Mechatronics, 2013, 18(2): 645-656.

[19] Ding X, Yu Y, Zhu J J. Trajectory linearization tracking control for dynamics of a multi-propeller and multifunction aerial robot-MMAR[C]//IEEE International Conference on Robotics and Automation (ICRA). 2011: 757-762.

[20] Sobral A, Vacavant A. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos [J]. Computer Vision and Image Understanding, 2014, 122: 4-21.

[21] Wang Y, Jodoin P M, Porikli F, et al. CDnet 2014: An expanded change detection benchmark dataset[C]// IEEE Conference on Computer Vision and Pattern Recognition Workshops. (CVPRW). 2014: 387-394.

[22] Wang Y, Luo Z, Jodoin P M. Interactive deep learning method for segmenting moving objects [J]. Pattern Recognition Letters, 2017, 96: 66-75.

[23] Sajid H, Cheung S C S. Universal multimode background subtraction [J]. IEEE Transactions on Image Processing. 2017, 26(7): 3249-3260.

[24] Allebosch G, Deboeverie F, Veelaert P, et al. EFIC: Edge based foreground background segmentation and interior classification for dynamic camera viewpoints[C]//International Conference on Advanced Concepts for Intelligent Vision Systems. Springer, Cham, 2015: 130-141.

[25] Sajid H, Cheung S C S. Background subtraction for static & moving camera[C]// IEEE International Conference on Image Processing (ICIP). 2015: 4530-4534.

[26] St-Charles P L, Bilodeau G A, Bergevin R. A self-adjusting approach to change detection based on background word consensus[C]// IEEE Winter Conference on Applications of Computer Vision (WACV). 2015: 990-997.

[27] Bianco S, Ciocca G, Schettini R. How far can you get by combining change detection algorithms? [C]//International Conference on Image Analysis and Processing. Springer, Cham, 2017: 96-107.

[28] Ramírez-Alonso G, Chacón-Murguía M I. Auto-adaptive parallel SOM architecture with a modular analysis for dynamic object segmentation in videos [J]. Neurocomputing, 2016, 175: 990-1000.

MOMDP solving algorithms comparison for safe path planning problems in urban environments

Jean-Alexis Delamer[1] and Yoko Watanabe[2] and Caroline P. Carvalho Chanel[3]

***Abstract*— This paper tackles a problem of UAV safe path planning in an urban environment where the onboard sensors can be unavailable such as GPS occlusion. The key idea is to perform UAV path planning along with its navigation an guidance mode planning where each of these modes uses different set of sensors and whose availability and performance are environment-dependent. It is supposed to have a-priori knowledge in a form of gaussians mixture maps of obstacles and sensors availabilities. These maps allow the use of an Extended Kalman Filter (EKF) to have an accurate state estimate. This paper proposes a planner model based on Mixed Observability Markov Decision Process (MOMDP) and EKF. It allows the planner to propagate such probability map information to the future path for choosing the best action minimizing the expected cost.**

I. Introduction

Safe navigation of autonomous vehicles in urban environment is a challenging problem. These vehicles rely on their onboard sensors to navigate through the environment. Their navigation performance depends directly on the onboard sensors whose availability and precision can vary with the environment. For example, the GPS localization precision depends on the satellites constellation and their visibilities. It is, however, possible to predict its localization precision, called Dilution of Precision (DOP), for a given environment [12]. Such information can be used as a priori knowledge in the path planning task, to ensure the safety under uncertainty.

In this context, this paper tackles such safe path planning problem for autonomous vehicles in urban environments, in particular for UAVs (Unmanned Aerial Vehicles). [18] and [1] have addressed UAV path planning problems, by considering the localization uncertainty which is propagated along a planned path in function of its environment. For instance, [18] applies the A* algorithm and makes use of uncertainty corridor to evaluate the plan for choosing the most efficient and safe path. [7] and [1] propagate the position uncertainty during path search by using RRBT algorithm. However, any of these approaches consider the complete GNC (Guidance, Navigation, and Control) closed-loop vehicle kinematics model into the decisional process.

The UAV safe path planning problem, addressed in this paper, is modeled as a particular Mixed-Observability Markov Decision Process (MOMDP) [16]. MOMDP is an extension of the classical Partially Observable Markov Decision Process (POMDP) [11], that allows the factorization of the state variables into fully and partially observable state variables. It holds in a smaller belief state space dimension, accelerating policy computation. The transition and observation functions of the MOMDP are built on the vehicle GNC model, and on the a priori knowledge of the environment given as probability grid maps of obstacles or sensor availabilities, respectively. Through these complex functions which combine continuous state variables transitions with discrete grid maps for sensor availability observations, the resulting updated belief state has a particular form. To address this difficulty, the belief state is approximated by a Gaussian Mixture Model (GMM) using the Expectation-Minimization (EM) algorithm [2].

Moreover, another particularity of this planning problem arises with the cost function proposed in this paper. It holds in having a non piecewise linear and convex (non-PWLC) value function [11], which prevents from using classical MOMDP solvers [16], [3]. Consequently, this paper presents two algorithms which do not require the PWLC property. The first algorithm is based on (L)RTDP (Labelled Real-Time Dynamic Programming) [5] and RTDP-bel [6] algorithms. (L)RTDP [5] improves the convergence of RTDP (and RTDP-bel in consequence) by labeling the already converged states. RTDP-bel use an hash-table only defined for belief states visited during policy learning. The second algorithm is based on POMCP [17], a Monte-Carlo tree search algorithm for partially observable environments. POMCP, as UCT (Upper Confidence bounds applied to Trees) [13], applies the UCB1 (Upper Confidence Bounds) greedy action selection strategy. POMCP approaches the value of a belief state by the average of evaluated costs during simulations which have started from this belief state, allowing this to generate a policy tree.And, as far as we know, RTDP-bel and POMCP are the only POMDP [11] algorithms that allows to approximate a value function in any format.

This paper is organized as follows: firstly the MOMDP model for this application case is presented. After, the belief state GMM representation learning is discussed. Then, the two algorithms: (L)RTDP-bel and POMCP-based are proposed; the results are shown in order to compare their performances. Finally, future work is discussed.

II. UAV Safe Path Planning Problem

This paper addresses a problem of finding a navigation and guidance strategy (path and modes) which makes

*This work was not supported by any organization

[1]Jean-Alexis Delamer, Information Processing and Systems Departement (DTIS), ONERA, Toulouse, France jean-alexis.delamer@onera.fr

[2]Yoko Watanabe, Information Processing and Systems Departement (DTIS), ONERA, Toulouse, France yoko.watanabe@onera.fr

[3]Caroline P. Carvalho Chanel, Design and Control of Aerospace Vehicles Departement (DCAS), ISAE-SUPAERO - University of Toulouse, France caroline.chanel@isae.fr

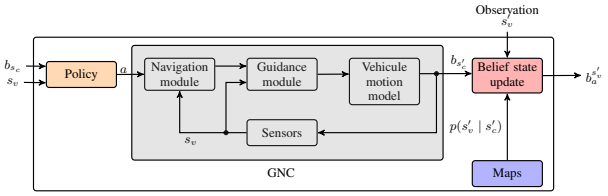


Fig. 1: System architecture diagram. The GNC closed-loop vehicle model is incorporated into the MOMDP transition function. Apriori information forms a set of probability grid maps of the environment and the sensor availabilities.

autonomous vehicles reach a given destination safely and efficiently in a cluttered environment. Autonomous vehicles are equipped with different sensors, such as INS and GPS, which are used in its GNC system to execute a path (Fig. 1). The work here presented can be applied to any type of autonomous vehicles, as long as their GNC model is well-defined. To illustrate the approach, this paper focuses on the UAV model proposed by [10], where the GNC closed-loop system is modeled as a transition function of the continuous vehicle state vector x (position, velocity, etc.). The navigation filter estimates this state x and its error covariance P for a selected navigation mode (or sensor). The guidance and control module executes a selected path by using (or not, depending on a selected guidance mode) the navigation solution. The execution precision is given by the execution error covariance Σ, which may depend on P. A priori knowledge on the environment is assumed to be given as a set of probability grid maps of obstacles and availability of each of the sensors. These maps are used in planning task to predict the path execution accuracy, and then to evaluate obstacle collision risk with respect to it given a path.

III. MOMDP Model

The Mixed Observability Markov Decision Process (MOMDP) proposed by [3] and [16] is a variant of the POMDP (Partially Observable Markov Decision Process). The state is not partially observable, but a part of the state is known at each epoch. In this problem, an UAV always knows the current sensor availabilities which are considered as a part of the state. Consequently, MOMDP is applied to model this problem. But, in contrast to a classical MOMDP model [16], there is no partial observable state in this application case. The vehicle state vector x is unobservable from the planning model point of view, since there is neither partial nor direct observation on it. The only outputs considered from the GNC closed-loop model is the localization and execution error covariances P and Σ. Figure 1 illustrates the system architecture with different modules.

The MOMDP is defined as a tuple $\{\mathcal{S}_v, \mathcal{S}_c, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0\}$, where $\mathcal{S}_v$ is the bounded set of fully observable states; $\mathcal{S}_c$ is the bounded set of non observable continuous states; $\mathcal{A}$ is the bounded set of actions; Ω is the bounded set of observations; $\mathcal{T}$ is the state transition function; $\mathcal{O}$ is the observation function such as : $\mathcal{O}(o, a, s'_c, s'_v) = p(o|s'_c, s'_v, a) = 1$ if $o = s'_v$, or 0 otherwise; $\mathcal{C} : \mathcal{B} \times \mathcal{B} \times A \rightarrow \mathbb{R}$ is the cost function, with $\mathcal{B}$, the belief state space defined over $|\mathcal{S}| = |\mathcal{S}_v| \times |\mathcal{S}_c|$; and $b_0 = (s^0_v, b^0_{\mathcal{S}_c})$, where $b^0_{\mathcal{S}_c} \in \mathcal{B}_c$ is the intial probability distribution over the non observable continuous states, conditioned to $s^0_v \in S_v$, the initial fully observable state.

The visible state $s_v \in \mathcal{S}_v$ is defined as a tuple containing the fully observable booleans of the sensor availability $[0; 1]$, with N the number of sensors, the boolean on the collision, and the P the localization error covariance matrix propagated by the navigation module in function of a selected navigation sensor/mode in a given decision step. The s_v is define such as $s_v = \{F_{\text{sensor1}}, \ldots, F_{\text{sensorN}}, F_{\text{Col}}, P\}$. It is assumed that the collision flag F_{Col} is observable either by measuring or estimating a force of contact.

The non observable continuous state $s_c \in S_c$ is defined such as $s_c = x$, recalling that x is the continuous vehicle state vector (position, velocity, etc.).

An action $a \in A$ is defined as a tuple $\{d, m_n, m_g\}$: the discretized path direction $d \in D$; the navigation mode $m_n \in \mathcal{M}_n$ and the guidance mode $m_g \in \mathcal{M}_g$.

The transition function $T(s_v, s'_v, a, s'_c, s_c)$ is composed of two functions: a transition function $T_{\mathcal{S}_c}$ such as:

$$T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) = f_{s'_c}(s'_c|s_c, s_v, a) \sim N(\bar{s'_c}, \Sigma'(s_v)),$$

which is based on the GNC closed-loop model, given that the probability distribution of a predicted state s'_c follows a normal distribution $N(\bar{s'_c}, \Sigma'(s_v))$, which in turn, is a function of the previous state s_c and the action a; and a transition function $T_{\mathcal{S}_v}$ such as $T_{\mathcal{S}_v}(s'_c, s'_v) = p(s'_v|s'_c)$, which represents the transition to s'_v and depends on the sensor availability maps and therefore depends only on the next state s'_c. Concretely,

$$T_{\mathcal{S}_v}(s'_v|s'_c) = \prod_{i=1}^{N+1} p(s'_v(i)|s'_c) \tag{1}$$

where N is the number of sensors, thus $N+1$ is the number of flags (booleans) in s_v, and $s'_v(i)$ the i-th flag. Then, the transition function becomes:

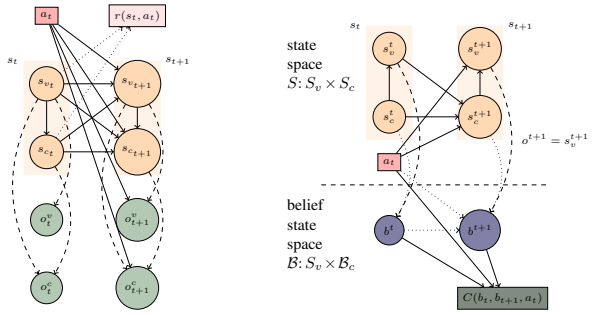
$$\begin{aligned} T(s_v, s'_v, a, s'_c, s_c) &= T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) \times T_{\mathcal{S}_v}(s'_c, s'_v) \\ &= p(s'_v|s'_c) f_{s'_c}(s'_c|s_c, s_v, a) \end{aligned} \tag{2}$$

Note the execution error covariance matrix Σ from the GNC transition model represents the uncertainty envelope of the unobservable state s_c (represented by b_{s_c} in the Fig. 1), instead of P, the localization error covariance matrix. The belief state is updated after an action a followed by a perceived visible state $o' = s'_v$. The belief state update is decomposed into two functions. The first one corresponds to the GNC closed-loop transition function belief propagation:

$$b_{s'_c}(s'_c) = \int_{\mathcal{S}_c} f_{s'_c}(s'_c|s_c, s_v, a) b_{s_c}(s_c) ds_c \tag{3}$$

The second one is the probability of s'_v (given by the probability grid maps) that is computed based on $b_{s'_c}$:

$$p(s'_v|b, a) = \sum_{i=1}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b, a) \tag{4}$$



(a) Classical transition model for a MOMDP [3]

(b) proposed MOMDP transition model with $\Omega = S_v$.

Fig. 2: Difference between the transition model

where c_i corresponding to the i^{th} cell of the probability map and $|G|$ is the number of cells in the map. Finally, the belief state update function is defined as :

$$b'^{s'_v}_{s'_c,a}(s'_c) = \frac{p(s'_v|s'_c)\int_{\mathcal{S}_c} f_{s'_c}(s'_c|s_c, s_v, a) b_{s_c}(s_c) ds_c}{\sum_{i=1}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b, a)} \quad (5)$$

a) Why a MOMDP model instead of a classical POMDP model?: The choice of modelling this application case as a MOMDP instead a POMDP (Partially Observable Markov Decision Process) is twofold: (i) the MOMDP model offers the possibility of factorizing the state space, resulting the policy computation complexity, as the belief state probability distribution can be defined over a small belief state space $\mathcal{B}_c$ (which refers to the $\mathcal{S}_c$ space instead of the complete $\mathcal{S}$ space, such that $b = (s_v, b_{\mathcal{S}_c})$) – see Fig. 2 ; (ii) as the state space is factorized, one can factorize the observation space too. In this particular application case, the observation set which corresponds to S_c is empty. Therefore, the only observation set relies on S_v. Given that $\mathcal{O}(o, a, s'_c, s'_v) = p(o|s'_c, s'_v, a) = 1$ iff $o = s'_v$, or 0 otherwise, one can compute expectations directly over this variable (cf. Eq. 4 and 5).

A. Belief state representation learning

Figure 3a illustrates a belief state update step. The future belief state b_a is calculated after an action a from an initial belief state b (see also Fig 1). If b is Gaussian, b_a returned by the GNC transition function is also Gaussian. Then, in this example, an observation s_v is perceived, such that a probability $p(s_v|b_a) > 0$ only in the blue shaded cells. By using this observation and (Eq. 5), the belief state b_a is updated to $b_a^{s_v}$. The shape of this new belief is no longer Gaussian. Consequently, future belief states will not be Gaussian neither.

Gaussian belief states allow algorithmic simplifications. The Gaussian property allows planning algorithms to handle belief states directly, unlike previous works which approach belief states with Monte-Carlo particle filters [4], [17], thus reduces computation time. Therefore, this paper proposes to apply a machine learning technique to calculate a Gaussian Mixture Model (GMM) to approximate the new belief $b_a^{s_v}$.

The EM algorithm [2] is used in this work. EM learns a GMM belief state representation for a fixed number N_g of Gaussian functions. The ideal N_g is unknown; then it is necessary to compare the GMMs learned for different N_g's. So firstly, a sufficient number of samples s_c is generated from $b_a^{s_v}$. Then, for each N_g, the GMM is *trained* (it runs the EM algorithm). To decide on which GMM best fits the belief state, the Bayes Information Criterion (BIC) [2] is used, due to its interesting property of penalizing the complexity of the learned model (i.e., N_g). The GMM with smallest BIC score is selected. Note that the maximum N_g and the number of samples are problem-dependant and need to be tested empirically beforehand. Figures 3b and 3c show an example of the GMM learning result, where the initial belief state b_a (black dotted ellipse) is updated by an observation s_v of non-collision (the white part of the figure 3b).Then EM is applied to learn GMMs with different N_g from 1 to 7, and that with $N_g = 2$ (shown in red and blue ellipses) was selected according to the BIC score comparison (Fig 3c).

B. Cost function

The GMM approximation allows us to analytically compute the cost $\mathcal{C}$ of the belief state transition. The cost function calculates the volume of the uncertainty corridor (see [18] for more details on this volume computation) between two belief states (ensuring the safety and efficiency of the path). A fixed collision penalty cost multiplied by collision probability is also added to the cost function. Then:

$$\mathcal{C}(b_t, b_{t+1}) = \sum_{g \in b_t} \sum_{g\prime \in b_{t+1}} U(g, g\prime) p(g\prime \mid b'^{s_v}_a, g) w(g) w(g\prime) + K \times s'_v(Collision)). \quad (6)$$

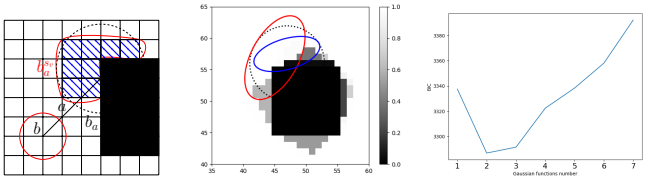
where b_t is the initial GMM belief, b_{t+1} is the learnt GMM belief, $U(g, g\prime)$ is the volume of the uncertainty corridor between two Gaussian functions from the mixtures, $p(g\prime \mid b'^{s_v}_a, g)$ the probability of being in $g\prime$ after an action a from g, $w(g)$ the weight of the Gaussian function provided by the EM algorithm and K the collision cost. Note the GMM approximation helps to compute the belief state transition cost (Eq. 6). The cost can be analytically computed, avoiding the application of particles filtering or Monte-Carlo cost evaluation.

C. Value function

The value function $V^\pi(b)$ is defined as the expected total cost (weighed by time using γ) the agent will receive from b_0 when following the policy π [11].

$$V^\pi(b) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, b_{t+1}, \pi(b_t))\right] | b_0 = b \right]. \quad (7)$$

As the value function is built based on an expected sum of costs, one needs to pay attention to the form of the cost



(a) Deformation of the b_a during update.

(b) GMM learning result with $N_g = 2$.

(c) BIC score for different N_g's

Fig. 3: Example of the GMM learning algorithm result.

function. Also note that the cost function does not directly depend on the action, but this last has an indirect impact: the uncertainty of a state depends on the execution error covariance Σ affected by the navigation and guidance modes chosen. The optimal policy π^* is defined by the optimal value function $V^{\pi *}$, such as :

$$V^{\pi^*}(b) = \min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, b_{t+1}, \pi(b_t))\right] | b_0 = b\right] \tag{8}$$

Opening the sum in (Eq. 8), it holds to a Bellman's equation, which allows the application of dynamic programming. For example:

$$\begin{aligned} V(b) &= \min_{a \in A} \mathbb{E}\left[\mathcal{C}(b, b_a^{s_v}) + \gamma V(b_a^{s_v})\right] \\ &= \min_{a \in A} \sum_{s_v \in S_v} p(s_v|b,a)(\mathcal{C}(b, b_a^{s_v})) + \gamma V(b_a^{s_v})), \end{aligned} \tag{9}$$

when the value (Eq. 9) converges for all reachable belief states, within an ϵ error, one can extract the related optimized (partial-)policy [14].

IV. Algorithms

The cost function defined in (Eq. 6) depends on belief state transitions and is no more piecewise linear and convex (PWLC). In this case, the use of classical MOMDP algorithms, such as SARSOP [14] which uses α-vectors to represent the value function, is no more possible.

A. (L)RTDP-bel

The proposed algorithm is based on (L)RTDP and RTDP-bel, because RTDP-like algortihms do not require to have a PWLC value function. The idea is to directly evaluate the belief states (and not the states as in RTDP-bel) while exploring the convergence improvement of (L)RTDP.

Therefore, some functions and definitions need to be adapted. In particular, the residual function which calculates the difference between the value of the belief state and the result of the Q-value of b for a greedy action a. Then the residual is defined as :

$$R(b) = \left| V(b) - \min_{a \in A} \sum_{s_v \in S_v} p(s_v|b,a)(\mathcal{C}(b, b_a^{s_v})) + \gamma V(b_a^{s_v}) \right| \tag{10}$$

As in (L)RTDP, it is considered that a belief state has converged (and consequently solved being marked as *Labelled*) if the following definition is verified.

A value function $V(b)$ converges for a given belief state b relative to parameter $\epsilon > 0$ when $R(b) \leq \epsilon$.

(L)RTDP-bel, shown in Alg. 1, takes in entry an initial belief state b_0 and an ϵ parameter. While the initial belief is not solved, it will continue to simulate the greedy policy (performing trials). A trial in (L)RTDP-bel is very similar to the one of (L)RTDP, but there is an important difference (besides working with belief states instead of the states): during the update of the belief state of (L)RTDP-bel, the EM algorithm is used to learn a Gaussian mixture model to represent the belief state. When the goal is reached[1] the algorithm checks if each of the value of the belief states has converged (i.e solved). This check is done by the *Check-Solved* algorithm which does not differ from the Check-Solved algorithm of (L)RTDP. To obtain more details on these algorithms, please refer to [5], [6].

[1]it considers that it has reached the goal when the position of the goal belongs to the ellipsoid (3σ) defined by the current belief state.

Algorithm 1: (L)RTDP-bel

```
1  Function (L)RTDP-BEL(b_0,ε)
2      while b_0 not solved do
3          (L)RTDP-BEL-TRIAL(b_0,ε)
4      return π*_{b_0}
5  Function (L)RTDP-BEL-TRIAL(b_0,ε)
6      visited ⟵ ∅; b ⟵ b_0
7      while b not solved do
8          visited ⟵ b
9          if b ∉ Goal then
10             a_best ⟵ argmin_{a∈A} Q^{V_l}(b,a)
11             V_l(b) ⟵ Q^{V_l}(b, a_best)
12             b_a ⟵ execute a_best in b
13             s_v ⟵ sample s_v from p(s_v|b_a)
14             b_a^{s_v} ⟵ update(b_a, s_v)
15             b ⟵ b_a^{s_v}
16     while visited ≠ ∅ do
17         b ⟵ pop(visited)
18         if !CHECK-SOLVED(b,ε) then
19             break
```

B. POMCP

The POMCP algorithm [17] is a Monte-Carlo Tree Search algorithm for partially observable environments. POMCP works by sampling a state s in the current belief state (belief node or history h) and simulating sequences of action-observation (rollout procedure) to construct a tree, then calculates the average reward (or cost) for a belief node based on the average reward of children nodes. The algorithm keeps in memory the number of times a node was explored $N(h)$ and the number of times an action was chosen $N(ha)$ in this given node. As UCT [13], it applies the UCB1 greedy action selection strategy that is based on a combination of two characteristics: an approximation of the action's Q-value and a measure (given by $c\sqrt{\frac{\log N(h)}{N(ha)}}$) of how well-explored the action is, given this history (or belief node).

However, due to the particularities of the model addressed in this work and to the fact that this is a goal-oriented problem, the POMCP algorithm needs to be modified. Starting with an initial belief state b_0 (line 3), the algorithm (Alg. 2) will expands the tree for a given timeout duration. If the belief state is the goal (line 6), it returns, else it tests if the belief state is in the tree. If not, (line 8) the belief state is added. For each pair of action-observation, the next belief $b_a^{s_v}$ is also added to the tree (line 11). Note that, contrary to the classical POMCP algorithm, no state is sampled because the algorithm works directly on the belief state (specially for cost and Q-value functions computation). Thus, the next action is chosen using the UCB1 greedy action selection strategy (line 12). After, an observation s_v is sampled, and the belief state is updated (EM algorithm). The tree is expanded with this new belief state, and the Q-value (recursively) updated.

Algorithm 2: POMPC

Function *POMPC(b_0, c, γ)*
  while *!Timeout* **do**
    Expand(b_0, c, γ)
  $a^* \leftarrow \underset{a \in A}{\operatorname{argmin}}\, V(b_0)$

Function *Expand(b, c, γ)*
  if $b \in$ *Goal* **then**
    return 0
  if $b \notin T$ **then**
    for $a \in A$ **do**
      for $s_v \in S_v$ **do**
        $T(b_a^{s_v}) \leftarrow (N_{\text{init}}(b_a^{s_v}), V_{\text{init}}(b_a^{s_v}), \emptyset)$
  $\bar{a} \leftarrow \underset{a \in A}{\operatorname{argmin}}\, Q(b,a) - c\sqrt{\frac{\log N(b)}{N(b_{\bar{a}})}}$
  $s_v \sim \mathcal{G}(b_{\bar{a}})$ /* Random generator */
  $b_a^{s_v} \leftarrow update(b_{\bar{a}}, s_v)$
  $Expand(b_{\bar{a}}^{s_v}, c, \gamma)$
  $N(b) \leftarrow N(b) + 1$
  $N(b_{\bar{a}}) \leftarrow N(b_{\bar{a}}) + 1$
  $Q(b,\bar{a})' \leftarrow \sum_{s_v \in S_v} p(s_v|b,\bar{a})\,(C(b, b_{\bar{a}}^{s_v}) + \gamma V(b_{\bar{a}}^{s_v}))$
  $Q(b,\bar{a}) \leftarrow Q(b,\bar{a}) + \frac{Q(b,\bar{a})' - Q(b,\bar{a})}{N(b_{\bar{a}})}$
  $V(b) \leftarrow \min_{a \in A} Q(b,a)$

C. Belief state value initialization

As the MOMDP value function results from the application of dynamic programming minimization operator, the expert needs to ensure that the initial value of a belief state (or initialization heuristic value, as in Alg. 2 with V_{init}) must be an upper bound (or lower bound for a maximization operation) in order to preserve algorithm convergence - in particular the contraction property [8].

The belief state value initialization proposed in this paper explores the A* shortest path solution on the obstacle grid map. The execution error Σ is propagated over this path for the navigation and guidance modes with sensors most likely-available. Then the cost-minimizing navigation and guidance mode is selected among all the available modes. This value approximation gives a tighter upper bound than a worst-case navigation and guidance strategy.

V. Simulation results

The two MOMDP algorithms have been tested on a benchmarking framework for UAV obstacle field navigation [2] [15], which provides environments with different obstacle configurations. The UAV model (is the one model described in Section III). Here two different maps have been selected: "Cube baffle" which contains two cube obstacles and "Cube" which contain one cube obstacle both with a grid size of $100 \times 100 \times 20$. To perform these tests, it has been considered only two sensors onboard an UAV: INS and GPS. While INS is known to be available anywhere, a probability grid map of GPS availability was created based on a DOP map calculated by using a GPS simulator. For each test the initial belief state was $b_0 = (\bar{s}_c, \Sigma, s_v)$, where $\bar{s}_c = [5,5,7,0,0,0,0,0,0]$, $\Sigma = diag(1,1,4,0.01,0.01,0.04,0.01,0.01,0.01)$, $s_v = [1,1,0,P]$, with $P = \Sigma$ (note this is true only on initial belief state, even after the first action Σ and P differs) and the goal was in $(90,90,7)$. To obtain representative results, 1000 simulations have been run for each algorithm policy on two different maps. The parameters for the (L)RTDP-bel were $\gamma = 0.9, K = 1000$ and for the POMCP policy : $\gamma = 0.9, Time = 180min, c = 200, K = 1000$.

[2]benckmark framework from: www.aem.umn.edu/people/mettler/projects/AFDD/AFFDwebpage.htm

	CubeBaffle		Cube	
	(L)RTDP-bel	POMCP	(L)RTDP-bel	POMCP
Success Rate	96%	95%	96%	96%
Average cost	3393.84	3072.67	5892.91	4093.24
Average cost in %	0%	-9.47%	0%	-30.54%

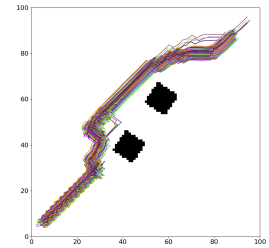
TABLE I: Performance comparison between (L)RTDP and POMCP.

The average simulation results are given in Table I. In terms of performance, the success rate is almost similar for each algorithm and map. However the average path cost is different between the algorithms, it can be seen that POMCP is more efficient than (L)RTDP. For the "Cube baffle" map the difference is smaller than for "Cube," POMCP reduces the cost of 9.45% for the first counter 30.54% to this last.

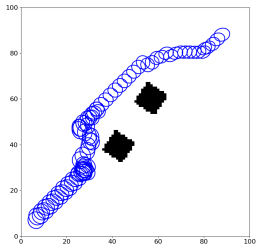
In Figure 4 some simulated paths are illustrated. The first column (Figures 4a, 4c, 4e and 4g) represent the simulated paths in 2D and the second column (figures 4b, 4d, 4f and 4h) represent the belief states of the most likely path. Regardless of the test map, the paths are perfectly within the bounds of the belief state calculated with the policies. The only exception is for the figures 4a and 4b, where different observations were perceived by the UAV during the simulations resulting in different paths. Thus for the sake of understanding only the belief states of the most likely path have been represented on the figure 4b. This supports the idea that representing belief states as GMM is a good idea to simplify cost computation and ensure the generalization of the MOMDP model with the GNC model.

Moreover, the paths simulated with the two algorithms are almost similar. This is because the test maps are simple and thus the shortest and safest path is straightforward to compute. However, it can be observed that the simulated trajectory with the POMCP policy is smoother. And more specifically with the "Cube" map, the POMCP has better-anticipated sensor availability and collision risk than (L)RTDP-bel. It was expected, because (L)RTDP-bel is greedier during policy computation and do more local optimization. (L)RTDP-bel optimizes better for belief states are closer to the obstacle than the POMCP which explores more uphill in the search tree. In the "Cube" map case the path simulated with the POMCP policy starts to avoid the obstacle much sooner resulting in lesser cost than the paths simulated with the (L)RTDP policy.

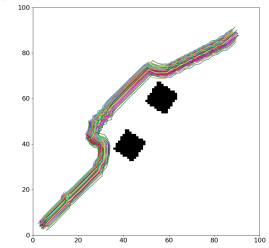
The 2D representation gives a good idea of the paths simulated, but it is interesting to analyze them in a 3D perspective. Figure 5 presents the simulated paths on the "Cube baffle" map Fig. 5a shows the paths obtained with the (L)RTDP-bel policy and Fig. 5b those with the POMCP policy. The first consideration is the paths are more scattered in height than in width. It is expected in these tests because GPS uncertainty is four time higher on height than on the other axes. It explains why the policies computed do not push the UAV to go above the obstacles because the maps are limited in height. Another consideration is that the POMCP



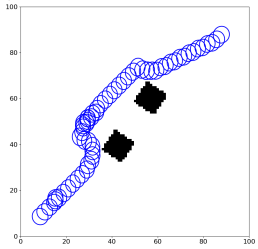
(a) Path simulated with the (L)RTDP policy on "Cube baffle".



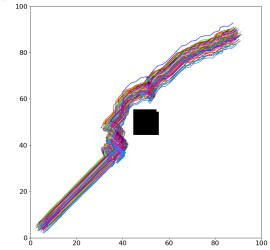
(b) Beliefs state representation of the (L)RTDP results.



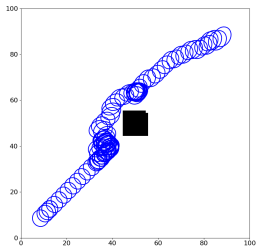
(c) Path simulated with the POMCP policy on "Cube baffle".



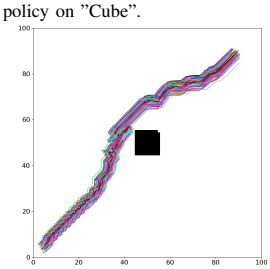
(d) Beliefs state representation of POMCP results.



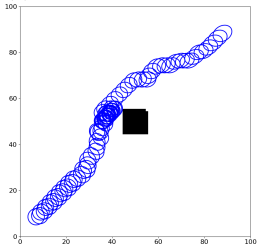
(e) Path simulated with the (L)RTDP policy on "Cube".



(f) Beliefs state representation of the (L)RTDP.



(g) Path simulated with the POMCP policy on "Cube".



(h) Beliefs state representation of POMCP results.

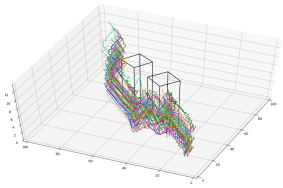
Fig. 4: Path simulated on the two maps for each algorithms.

policy (Fig. 5b) anticipate uncertainty around the obstacle by avoiding it and getting high.

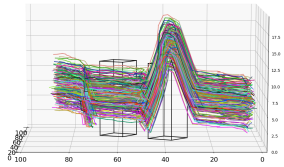
From these results, it is possible to say that both algorithms compute an acceptable policy. But POMCP has calculated a more effective policy due to its better convergence properties [17].

VI. Conclusion and Future Work

This paper presented a MOMDP model to solve safe path planning problem in urban environments, a belief state representation as Gaussian Mixture Models is presented, two algorithms are proposed, and results are compared. This problem can be viewed as a large MOMDP domain, as the non-observable state is continuous. Even when considering a discrete set of actions the problem is complex. The current results show that with goal-oriented algorithms it is possible to obtain significant results on simple maps. More evaluations are necessary, especially on real urban environments



(a) Path simulated with the (L)RTDP policy on "Cube baffle".



(b) Path simulated with the POMCP policy on "Cube baffle".

Fig. 5: 3D representing of the path simulated with each policy on the "Cube baffle" map.

[15].

Further work will include an extension to deal with a continuous action space. It is also planned to apply learning algorithms for value function and policy representation. So that, one can generalize the value (or policy) for belief states not visited during the optimization phase. Gaussian Process Dynamic Programming [9], for example, could be a potential solution to this interesting point.

References

[1] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart. Motion- and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics*, 2014.

[2] D. W. Andrews and B. Lu. Consistent model and moment selection procedures for gmm estimation with application to dynamic panel data models. *Journal of Econometrics*, 2001.

[3] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet. A closer look at momdps. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2010.

[4] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo. Monte carlo value iteration for continuous-state pomdps. In *Workshop on the algorithmic foundations of robotics*, 2010.

[5] B. Bonet and H. Geffner. Labeled rtdp: Improving the convergence of real-time dynamic programming. In *ICAPS*, 2003.

[6] B. Bonet and H. Geffner. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, 2009.

[7] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] O. Buffet and O. Sigaud. Processus décisionnels de markov en intelligence artificielle, 2008.

[9] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 2009.

[10] J.-A. Delamer, Y. Watanabe, and C. P. C. Chanel. Towards a momdp model for uav safe path planning in urban environment. In *IMAV 2017*, 2017.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

[12] F. Kleijer, D. Odijk, and E. Verbree. Prediction of gnss availability and accuracy in urban environments case study schiphol airport. In *Location Based Services and TeleCartography II*. Springer, 2009.

[13] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML*, 2006.

[14] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[15] B. Mettler, Z. Kong, C. Goerzen, and M. Whalley. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. In *66th Forum of the American Helicopter Society:" Rising to New Heights in Vertical Lift Technology", AHS Forum 66*, 2010.

[16] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 2010.

[17] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, 2010.

[18] Y. Watanabe, S. Dessus, and S. Fabiani. Safe path planning with localization uncertainty for urban operation of vtol uav. In *AHS Annual Forum*, 2014.

On finding low complexity heuristics for path planning in safety relevant applications

Krutsch Robert, Intel

***Abstract*— In many robotics applications path planning has safety implications that need to be addressed and understood. Approaches based purely on learning algorithms have today no strong guarantees that the path found, even given perfect environment model, is safe. In contrast, search based methods have strong theoretical guarantees but are significantly slower and hard to parallelize. In this paper we present a method of obtaining heuristics for search based algorithms targeting to reduce the search complexity by combining the strengths of the two paradigms. We show that a complexity reduction of more than 30% is achievable with less than 1% drop in path optimality. As a consequence of the complexity reduction we also measure a performance boost of more than 30%.**

I. INTRODUCTION

Comprehensive environment models are an essential part of robotics application and driver assistance. A significant number of publications [1,2,3,4] have concentrated on developing strategies for producing predictive environment models that incorporate information from multiple sensors. The typical approach in robotics has been based on the sense – plan – act paradigm, where environment models constructed based on the sensed information are used in the planning phase. Other paradigms, based on reactive control need also a representation of the environment yet concentrate more on reaction timeliness than on environment model correctness [5].

Path planning has been studied extensively in the past years and many of the algorithms have been implemented and tested in various robotics applications [6]. Most of the modern strategies perform well in environments where no adversarial interaction occur or/and where sensing is still acceptable accurate. In driver assistance systems for instance path planning based on a grid based representation of the environment is very often used. Such a representation allows the designer to choose from multiple methods of path planning as for instance probabilistic roadmap, rapidly exploring random trees, search based methods (e.g. variants of A*) , reinforcement learning methods etc.

In automotive application functional safety is one of the critical aspects needed to be considered in system design. The ISO26262 [7] gives guidelines and requirements at the system level, defining the framework on how hardware and software is built. The framework does not mandate specific implementation features, so the designer has the freedom and the responsibility of how to achieve the functional safety for the final product. Typically, the system safety goals are achieved by decomposing the system and applying redundancy and fault detection mechanisms. In such systems multiple environment models are used, different modalities for path planning are employed to achieve the system level safety goals. One possible architecture is presented in Figure 1, where two different environment models are used, different possible paths are obtained and then the cross check between environment paths is performed. Note that typically a significant number of paths is produced by each path planner (e.g. 100), with different characteristics and after validating this paths one both environment models one path will be selected for the execution.

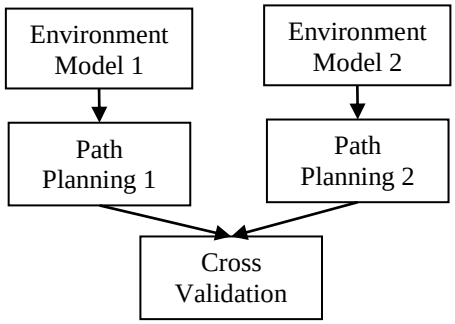


Figure 1: Functional safety decomposition of the planning and environment modeling

The algorithmic diversity is a very important part of the architecture that allows such a decomposition to be valid. For instance, different methods of path planning can be used or different heuristics can be employed. For the environment model different settings can be used such that one environment model has a longer aggregation time while the other is more refined for quick environment changes.

In this paper we will concentrate mainly on search based path planning and propose methods on how to reduce the search complexity by producing search heuristics based on convolutional neural networks. We will study also the tradeoff between complexity reduction and optimality. It is expectable that such heuristics might louse the optimality guarantees yet in such systems the optimality is in itself very hard to quantify. For autonomous driving applications the optimality does not only imply the shortest path but might be formulated as a combination of shortest path, lowest energy consumption and comfort.

Due to the increased number of paths that need to be found a reduction in search complexity is highly desirable. Typically neural networks run best on accelerators while search based schemes have better run times on cores

allowing for a pipelining of producing heuristics and searching a path as depicted in Figure 2. To be noted is also that if on the target hardware multiple neural network compute accelerators and cores are available the process can be parallelized and so provide more diversity to the solution and potentially faster runtime.

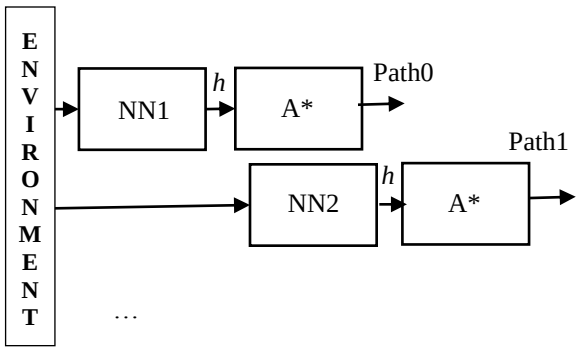


Figure 2: Pipelining the neural network execution with the A* search. Each neural network (NN1, NN2...) produces a new heuristic denoted as *h* that is used by A* to produce the paths Path0, Path1 etc.

The search scheme based on A* has strong guarantees since it test each cell if it is occupied or not and is as such guaranteed not to have in the path produced any obstacle, as long as the environment model is accurate. Mitigation of the sensorial imperfections and adversarial behaviors is typically based on continuous and timely replanting. The practical assumption is that the sensors get more accurate as the robot is approaching the obstacle and that the physical constraints of the other agents prohibits them from changing strategies with a very low time granularity (the behavior is stationary in the re-planning interval). This assumptions are reasonable for most use-cases and can be fairly accurate for instance in parking use-cases. Approaches such in [7] have no guarantee on correctness of the path and generally reinforcement learning approaches are very susceptible to catastrophic forgetting and can be used in practice only after a validation step where correctness and physical constraints are taken into consideration.

II. Experimental Setup

For our experiments we have constructed a dataset of 500000 occupancy grids of size 32x32 where each cell is marked either as free or as occupied (the probability of a cell to be occupied is 0.4, an example is provided in Figure 3). The starting location and the goal are encoded also as a grid with one in the respective locations and zero in the rest. An agent is placed in the start location and is able to take an action from the action set *A = {left, right, up, down, left-up, right-up, left-down, right-down}*. A move up, down, left or right is associated with a cost of one while a move diagonally has a cost of 1.41. The path cost is the summation of the cost of the individual moves. The data set contains also the Euclidian distance heuristic encoded as grid and the optimal heuristic that is found after running the A* algorithm from every location to the goal. As can be expected, computing the optimal heuristic is the most demanding part in the dataset generation and takes a few weeks on our system setup. We also keep the number of opened nodes by A* for the Euclidian distance heuristic.

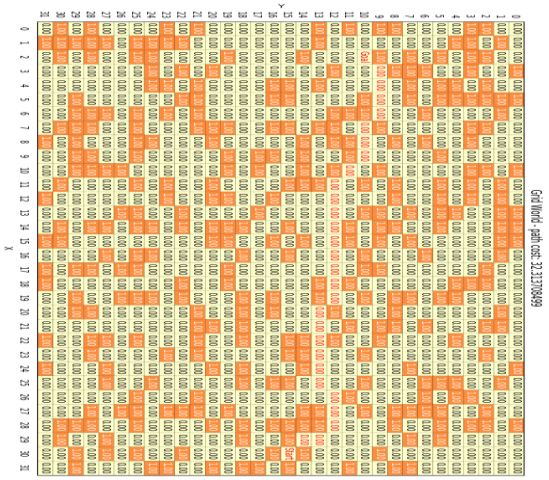


Figure 3: Occupied grid cells are marked with orange, the path from start to goal is marked with light orange

Comparing the complexity associated with two heuristics implies comparing the number of open nodes during the A* search. The second dimension of interest is the optimality of the path; it is well known that in case the heuristic overestimates the cost to the goal there are no optimality guarantees. We have used the following two performance criteria to assess the quality of the new heuristic:

$$C[\%] = \frac{\sum(OE-ONN)}{\sum OE} * 100 \quad (1)$$

$$O[\%] = \frac{\sum(DNN-OD)}{\sum OD} * 100 \quad (2)$$

, WHERE:

OE = Number of opened nodes by A* given Euclidian distance as heuristic
ONN = Number of opened nodes by A* given output of the neural network as heuristic
OD = optimal path length
DNN = path length found by A* given the heuristic obtained with a neural network
The summation if (1) and (2) are over the test dataset.

In our experiments we obtain new heuristics with a neural network that has as input three channels (Figure 4):

- Start – Goal channel that has the same dimensions as the grid with one in the start and goal location and zero everywhere else
- Grid channel that has one in every cell where an obstacle is present and zero where no obstacle is present
- Euclidian distance channel that has the Euclidian distance from each cell to the goal. This channel can be obtained in an offline computation.

The network output is a matrix of the same size as the grid in that each element represents an estimate of the path length from that specific location to the goal. This matrix is used as

a heuristic for A* and used to compute the performance metrics presented in (1) and (2).

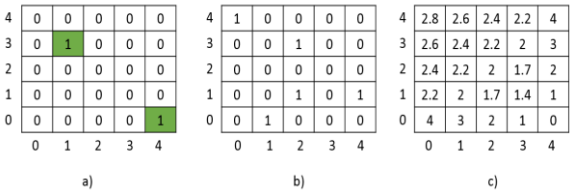


Figure 4: Input channels to the neural network: a) Start-Stop channel, b) Grid Channel, c) Euclidian distance channel

III. Obtaining new heuristics

The goal is to obtain a new heuristic that is less complex and that loses as less as possible from the optimality characteristic. Complexity and optimality cannot be obtained directly by a differential function that can be introduced in the neural network so we will concentrate on finding cost functions that are correlated to this indicators such that optimizing this cost functions will improve the two hidden indicators.

In [8] Theorem 12 from Chapter 3 indicates that given two admissible heuristics $h_2(n)$ and $h_1(n)$ for that $h_2(n) \geq h_1(n)$ for every *n* imply that an A* search using h_2 is more efficient than A* search using h_1 (this is intuitively obvious since we have a better approximate of the optimal heuristic). It is also shown in [8], Chapter 7, that when non-admissible heuristics are used the accuracy with that we approximate the optimal path is not the most critical aspect since less accurate estimations can lead to more informed heuristics (this is somehow surprising but has as background the idea that overestimates outside the path can be benefic to the search complexity). Given the insides above and the fact that the neural network is not guaranteed to produce an admissible heuristic it is clear that a simple formulation of the cost function where we try to approximate as much as possible the optimal path is prone to produce uncorrelated results to the complexity and optimality metrics.

In our experiments we test the following cost functions:

$$L_1 = \frac{|h_o - h|}{|h_e - h|} \quad (3)$$

$$L_2 = |h_o - h| + Relu(|h_e - h|) \quad (4)$$

, where:

h_o- The optimal heuristic, true distance to the goal

h - Output of the neural network

h_e - Euclidian distance heuristic

The denominator in L_1 encourages the output of the neural network to be greater than the initial Euclidian heuristic while the numerator in L_1 is responsible for guiding the network towards the optimal policy and penalizes severe overestimates. We use in the computation of the loss all cells that are not occupied and give equal weighting to cells that are on path and off path.

L_2 is constructed out of two terms , the first term encourages the output of the neural network to be close to the optimal heuristic while the second term pushes the output to be higher than the Euclidian distance heuristic.

We have experimented also with several other loss function formulations where the terms of the loss function are weighted or where we treat the loss differently if it is off the minimal path or on the minimal path (inspired by the fact that overestimate on path can be highly detrimental to complexity). The results obtained where similar to the vanilla formulation from (3) and (4) but the training procedure was observed to be more unstable.

The neural network architecture used in our experiments is depicted in Figure 5 and has a contracting part and an expending part similar to topologies employed in pixel labeling applications. The intuition behind comes from [9] where the grid is split into blocks and the planning is done hierarchically, first between blocks and then inside blocks. Similarly, the network contracts the information into blocks given by the receptive field of the convolution and pooling and in the later stage incorporates more fine grained information. The parameters for the topology are detailed in Table 1. As it is custom we have split the data set into training, validation and test and optimized the network with the Adam optimizer with learning rate 1e-4. The batch size used throughout the experiments is 128.

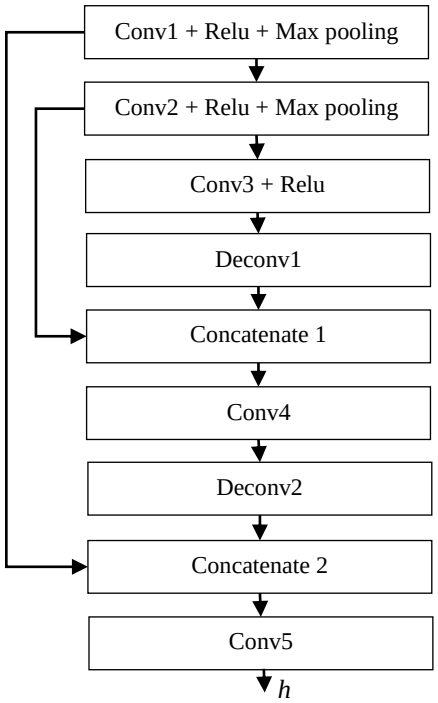


Figure 5: Neural network topology used in the experiments. Conv1:5 are two dimensional convolution layers while Deconv1:2 are two dimensional transpose convolutions

Layer	Kernel Size	Stride	Out Channels
Input	-	-	3
Conv1+Relu	3x3	1	32
Max Pooling	3x3	2	32
Conv2+Relu	3x3	2	16
Max Pooling	3x3	2	32

Conv3+Relu	3x3	2	8
Deconv1	3x3	2	8
Concatenate 1	-	-	24
Conv4	3x3	1	8
Deconv2	3x3	2	8
Concatenate 2	-	-	40
Conv5	3x3	1	1

Table 1: Neural network parameters

IV. Results

One of the first questions that arise is if the loss functions are correlated with the algorithmic complexity and optimality of the A* algorithm. In Figure 6 and 7 the training process statistics are shown (last 105 epochs of the training), the correlation between (1) , (2) and the L_1 and respectively L_2 loss functions are immediate to spot since as the loss decreases the complexity reduction has an increasing trend while the loss in optimality decreases. The metrics of (1) and (2) are computed over the validation test that represents about 5% of the overall dataset.

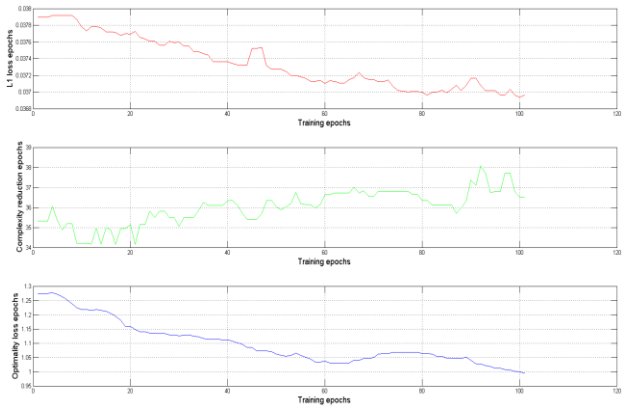


Figure 6: Upper plot is the L_1 loss; middle plot the complexity reduction (1); lower plot the optimality lost (2). The x axis for all plots represents the training epoch

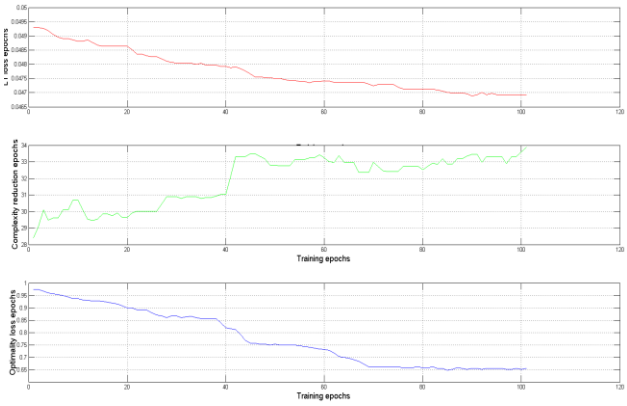


Figure 7: Upper plot is the L_2 loss; middle plot the complexity reduction (1); lower plot the optimality lost (2). The x axis for all plots represents the training epoch

The metrics (1) and (2) for the two loss functions are presented in Table 2 and are obtained based on the test subset that has 128000 grid samples. The computational time gain as measured on our system is presented in the 3rd column of the Table 2 and it is based on the formula (5). As can be observed the computational time reduction follows closely the complexity reduction. In our experiments we run first the A* algorithm for the heuristic obtained with the neural network and after the search based on the Euclidian distance heuristic. While the data cache is flushed in between the runs of A* the instruction cache is not flushed making it highly likely that the run of the Euclidian distance based search benefits from a better hit ratio, nevertheless the runtime is reduced by more than a 3rd.

$$T[\%] = \frac{\sum(T_{old} - T_{new})}{\sum T_{old}} * 100 \quad (5)$$

, where:
T_{old} – Runtime for the Euclidian distance heuristic A*
T_{old} – Runtime for the neural network heuristic A*

Loss Function	C[%]	O[%]	T[%]
L_1	37.64	1.01	36.32
L_2	35.82	0.71	32.45

Table 2: Performance metrics for L_1 and L_2

V. Conclusion

In this paper we have shown a novel method of obtaining new heuristics for A* that have lower complexity than the starting Euclidian distance with a minimal loss in complexity. The loss functions used follow the intuitions obtained from theoretical results obtained in [8]. We show that such indirect cost functions are correlated to the hidden variables in the A* algorithm and that joint optimization is possible.

Future work will investigate higher dimensional graphs and introduce robot poses in the search process.

References

[1] Florian Homm et al.,Efficient Occupancy Grid Computation on the GPU with Lidar and Radar for Road Boundary Detection. IEEE Intelligent Vehicles Symposium, 2010, San Diego, 10.1109/IVS.2010.5548091.

[2] Dominik Nuss, Doctoral Dissertation: A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-Time Applications, 2016, University Ulm

[3] Alberto Elfs et al., Using Occupancy Grids for Mobile Robot Perception and Navigation, IEEE Computer Volume:22 Issue:6, 1989, 10.1109/2.30720

[4] Sebastian Thrun. Learning Occupancy Grids with Forward Models. International Conference of Intelligent Robots and Systems, 2001, Maui, 10.1109/IROS.2001.977219

[5] Daniel Kappler et al. Real time perception meets Reactive Motion Generation , IEEE Robotics and Automation Latters, 2018, 10.1109/LRA.2018.2795645

[6] Eric Galcern et al. A survey on coverage path planning for robotics Robotics and Autonomous Systems 61 1258-1276, 2013

[7] Aviv Tamar et al. ,Value Iteration Networks, NIPS , 2016

[8] Pearl, Judea. Heuristics: Intelligent Search Strategies for Computer Problem Solving , Michigan, Addison-Wesley, 1984, Chapter 3,7

[9] Adi Botea et al. Near Optimal Hierarchical Path-Finding, Journal of Game Development, Volume 1, Pages 7-28, 2004

Enhancing the educational process related to autonomous driving

Nikos Sarantinoudis[1], Polychronis Spanoudakis[2], Lefteris Doitsidis[3],
Theodoros Stefanoulis[2], Nikos Tsourveloudis[2]

***Abstract*— Autonomous driving is one of the major areas of interest for the automotive industry. This constantly evolving field requires the involvement of a wide range of engineers with complementary skills. The education of these engineers is a key issue for the further development of the field. Currently in the engineering curriculums, there is a lack of related platforms that can assist the engineers to train in and further develop the required dexterities. The current practice is using either small robotic devices or full scale prototypes in order to understand and experimentate in autonomous driving principals. Each approach has disadvantages ranging from the lack of realistic conditions to the cost of the devices that are used. In this paper we present a low cost modular platform which can be used for experimentation and research in the area of autonomous cars and driving. The functionality of the suggested system is verified by extensive experimentation in - very close to- real traffic conditions.**

I. INTRODUCTION

In recent years, autonomous driving has emerged as a major innovation in the automotive industry. Currently the technology has matured and commercialisation is expected in the following years. Autonomous driving, refers to the ability of a vehicle to perceive its surroundings with various attached sensors, evaluate the conditions and navigate to an exact location safely without the interference of a human driver, taking into consideration the ever-changing and unpredictable environment. Due to the broadness of the term autonomy, SAE International has explicitly defined the levels of automation to characterize the extent to which a vehicle can drive autonomously [1], [2]. Explaining autonomy further, like every robotic system, autonomous vehicles utilize the "sense-plan-act" design. They use a combination of sensors to perceive the environment including but not limited to lidar (light detection and ranging), radar, cameras, ultrasonic and infrared. The information gathered is fused and used for decision making. As far as it concerns localization a combination of the Global Positioning System (GPS) and Inertia Measurment Units (IMU) is currently used [3].

Autonomous driving has significant internal (user) and external (external actors) impact. It provides a stressful environment for the driver and increases its productivity during the time spend inside the car, as well as it provides the option to non-drivers (or incapable to drive people) to commute. On the external side, increased safety (reducing crash risks and high-risk driving), increased road capacity and reduced costs, increased fuel efficiency and reduced pollution are equally important [4].

Even though the related technology has made significant advances on the domain, the cost of developing and experimenting with autonomous cars still remains high. Apart from the standard platform cost, there is a significant cost related with the actual sensors and processing unit which are essential for autonomous capabilities.

The aforementioned cost makes it difficult for a platform to be adopted in the educational and research activities of higher education institutes. There is always the option of small autonomous robotic vehicles [5], [6], [7], [8], [9] which are adequate for understanding the basic concepts of autonomous driving, or even inexpensive RC-car based autonomous car testbeds but few attempts have been made to provide low cost realistic platforms for research and education. A detailed comparison between our approach and the other available vehicles is presented in Table I.

TABLE I
COMPARISON OF VARIOUS EDUCATIONAL PLATFORMS

Comparison of Educational Platforms						
Platform	Camera	LiDAR	Ultrasonic	GPS	IMU	Scale
Duckietown	✓	✗	✗	✗	✗	Radio Conrtrolled
Donkey Car	✓	✗	✗	✗	✗	Radio Controlled
BARC	✓	✗	✗	✗	✓	Radio Controlled
MIT Racecar	✓	✓	✗	✗	✓	Radio Controlled
F1/10	✓	✓	✗	✗	✓	Radio Controlled
Tucer	✓	✓	✓	✓	✓	Urban Mobility Vehicle

This paper proposes an approach that will allow the educational process on the domain of autonomous driving to become viable and inexpensive, providing students the ability to apply the principles related to autonomous driving in a realistic low cost platform using tools that will minimise the development time and maximise the efficiency, gaining important hands-on experience. The platform is based on a single seater, urban concept vehicle, that has previously competed in Shell Eco Marathon Europe competition [10], with TUCer Team [11] from Technical University of Crete (TUC). The aforementioned platform has been in constant development and has been used as the testbed for research in automotive engineering [12], [13], [14], [15]. It has been modified from a hydrogen fuel cell powered car to a battery powered autonomous vehicle. In order to achieve this goal a series of hardware and software solutions were adopted

[1]N. Sarantinoudis is with the School of Electrical & Computer Engineering, Technical University of Crete, University Campus, Chania GR-73100, Greece
nsarantinoudis at isc.tuc.gr
[2]P. Spanoudakis, T. Stefanoulis and N. Tsourveloudis are with the School of Production Engineering and Management, Technical University of Crete, University Campus, Chania GR-73100, Greece
hronnis, nikost at dpem.tuc.gr
[3]L. Doitsidis is with the Department of Electronic Engineering, Technological Educational Institute of Crete, Romanou 3, Chania GR-73133, Greece
ldoitsidis at chania.teicrete.gr

and several devices were installed. Stepper motors have been fitted for steering and braking control as well as various sensors for perception (Stereo Camera, Lidar, Ultrasonic) and localization (GPS, IMU) along with an embedded computer. All of the above sensors are off-the-shelf components, easy to acquire and use and suitable for entry-level approach on the autonomous vehicle domain. Our architecture is simple, yet efficient, and the combination of Nvidia's Jetson TX2 (main processing unit) along with a set of microcontrollers can support from plain digital and analog inputs to even CAN (automotive standard) connectivity. More details will be provided in Section II. The functionality of the proposed approach is highlighted through simple yet realistic experiments that highlight the efficiency of our approach.

The rest of the paper is organised as follows, in section II the testbed is described in detail including the powertrain, the steering and braking system, the processing unit, the sensors used as long as the software sollutions adopted. In section III experimental test cases are presented that proves the functionality of the proposed approach. Finally in section IV concluding remarks and future directions for research and development are presented.

II. TESTBED

The testing platform is based on a single-seater custom vehicle, designed, developed and manufactured at the Technical University of Crete. The chassis of the car consists of aluminum tubing and the cover is made from carbon fiber. Its dimensions measure 2.5 x 1.25 x 1m (L x W x H) and its curb weight 108 kg. In Fig. 1 the prototype vehicle, and the position of all the major components is depicted, while in Fig. 2 we present all the major electronic components and the motor of the vehicle. In this section we will describe in detail the proposed testbed and its core components.

Fig. 1. The prototype autonomous vehicle

A. Powertrain

The vehicle is equipped with PGM132, a permanent magnet direct current brushed motor from Heinzmann. At 24V it outputs 1.8kW at 1100 rpm with rated torque of 15Nm (peak 38Nm) and 90A current. This motor is capable of inputs up to 60V, raising power to 5.1kW adding to our platform various performance profiles. The motor is controlled by an Alltrax

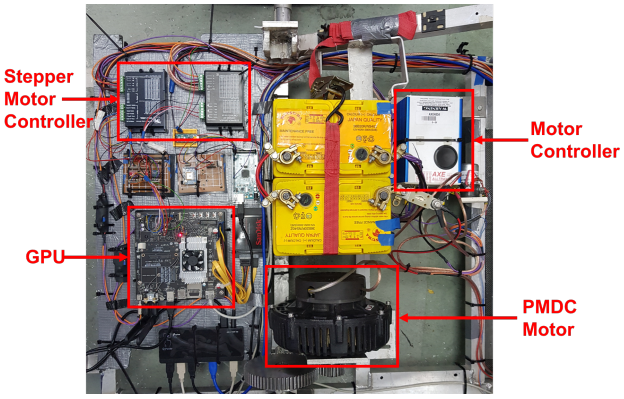


Fig. 2. The electronics and the motor of the testing platform

AXE 4834 permanent magnet motor controller delivering 135Amps rated and 300Amps peak, more than enough for the motor paired with.

The propulsion system is supplied by two 12V-40Ah rechargeable lead-acid batteries, providing adequate power for propulsion to urban mobility speeds (∼50km/h) with sufficient range (for testing purposes) before recharging. Additionally, a separate battery pack, consisting by 2 12V-7.2Ah rechargeable lead acid batteries supplies the various sensors and the on-board computer as described in II-C and II-D respectively. The reason for the separate supplies, even though the voltage is the same, lies to the fact that the common supply of motors and electronic devices, adds substantial electromagnetic noise to the circuit disrupting the integrity of sensor readings. Furthermore, motor's inrush current or sudden load changes (acceleration or hill climbing) could potentially cause voltage drops disrupting the constant power application needed by the electronic devices on board the vehicle. The power from the motor is transmitted to the wheels via a fixed gear ratio with one gear directly attached to the motor's rotating axle and the other to the wheel axle.

B. Steering and Braking System

Steering and braking control has been utilized with stepper motors, incorporated in the already existing steering mechanism and brake pedal, providing the ability to the driver to take control in a moments notice quiting the autonomous navigation if necessary or do not engage it at all and drive the vehicle manually. Both braking and steering stepper motors are powered by the propulsion batteries at 24V and controlled from Jetson TX2 GPIO ports.

The steering mechanism on this car is an implementation of the Ackerman steering geometry. A stepper motor is placed on the steering rack, using a pair of gears with fixed gear ratio rotating it directly to the desired direction. The hybrid stepper motor from Motech Motors with 1.26Nm of torque and step of 1.8^o is controlled by a Wantai Microstepping Driver. The driver is set up to further enhance the precision of the motor resulting in steps of just 0.225^o. In the end of the steering rack, a rotary magnetic modular

encoder is attached providing information about steering's exact position.

The vehicle is equipped with Shimano hydraulic bike pistons and disk brakes on the wheels. The braking control utilizes a similar design. A stepper motor with 4Nm of torque and 1.8^o steps actuate the brake pedal. A microstepping driver is set to the same level of precision for ease of programming and coherence between braking and steering steps. Braking stepper motor packs higher torque due to the needs of the lever pulling design for brake actuating.

C. Processing Unit

The central processing unit installed on the vehicle is an NVIDIA Jetson TX2 Developer Kit. It is an embedded ARM architecture equipped with a Quad ARM A57 processor plus a Dual Denver processor, 8GB of LPDDR4 memory and 256 CUDA cores of NVIDIA's Pascal Architecture, able to deal with the most computational intensive processes. It supports CAN, UART, SPI, I2C, I2S protocols as well as plain GPIO. For higher level communication WLAN, Gigabit Ethernet and Bluetooth is supported together with USB 3.0 and USB 2.0 ports. Internal memory is 32GB eMMC, but SATA connectivity and SD Card port are available too. In our setup the a 240 GB Sandisk Extreme II Solid State Disk is used for speed and extended storage capabilities. Additionally, a TP-Link 7-port powered USB 3.0 hub is attached to the USB 3.0 port, for easier multiple peripherals connectivity and uninterrupted power supply, since built-in ports are able to deliver 900mA of current, not enough for the attached USB devices. The Jetson TX2 was selected for two main reasons: (i) the extended capabilities that provides for a reasonably low cost and (ii) the ability to rapidly prototype solutions and test them in a the real testbed using standard tools which are widely used in the education process (i.e. Maltab, libraries provided by NVIDIA or other open resources etc.).

D. Sensors

A wide range of sensors for perception and localization have been used. The overall set-up is presented in Fig. 1. As far as it concerns the perception sensors, a ZED stereo camera from Stereolabs is mounted at the center of the vehicle. It consists of two 4 Megapixels cameras, with wide-angle dual lenses, field of view at 90^ox60^ox110^o (HxVxD) and f/2.0 aperture. Sensors are 1/3" with backside illumination capable of high low-light sensitivity and have a native 16:9 format for a greater horizontal field of view. USB 3.0 connectivity allows for high resolutions (up to 2.2K@15 FPS). Equally important are the depth recognition capabilities of the ZED camera, achieving the same resolution as the video with range from $0.5m$ to $20m$ and the motion sensing with 6-axis Pose Accuracy (Position:+/- $1mm$ and orientation: 0.1^o) using real-time depth-based visual odometry and SLAM (Simultaneous Localization and Mapping). Our choice was further backed by the fact that Sterolabs provide an SDK for the Jetson TX2, our main processing unit.

The second sensor responsible for perceiving the environment is a Scanse Sweep lidar. It consists of a LIDAR-lite v3 sensor from Garmin, in a rotating head, thus providing 360^o scans of the surroundings. It has a range of $40m$ and a sample rate of 1000 samples per second. It scans on a single plain (2 dimension lidar scanning) and is suitable for Robotics and UAVs. The counterclockwise rotating head emits a beam with $12.7mm$ diameter which expands by approximately 0.5^o. Scanse Sweep pairs it with and SDK for easy implementation to projects, as well as a visualizer app for graphical representation of the measurements.

Apart from the aforementioned sensors there are also forward mounted, two Maxbotix MB1010 Lv-MaxSonar-EZ1 ultrasonic sensors. With a range from $15cm$ to $645cm$ and a refresh rate of $20Hz$, they are capable of accurately detect and measure the distance from objects in front of the vehicle. They have multiple connectivity protocols such as analog, pulse width and serial output for easier sensor integration to different platforms. Due to cross-talk issues between the two sensors, a chaining between them is required, triggering sensors one after another eliminating the interference on the measurements of each individual sensor.

Both the Stereo Camera and the Lidar are connected to the USB hub and then directly to Jetson TX2. For the ultrasonic sensors, the analog output is selected to be used. Since Jetson TX2 has no analog inputs, a microcontroller is used as middleware between the sensors and the processing unit improving input-output capabilities of the system.

For localization purposes an Adafruit Ultimate GPS Version 3 is used. Capable of using up to 66 channels and having -165dBm sensitivity, the achieved accuracy is adequate to position our vehicle in the real world with pinpoint accuracy. For more robustness and better satellite reception, an external GPS antenna is used (instead of the built-in) with the u.FL connector provided in this board. The serial output of the GPS sensor, similarly to the ultrasonic sensors, uses a microcontroller as a middleware between the device and the processing unit. Complementing the GPS an IMU module is also used. The Sparkfun Razor IMU is a 9 Degrees-of-Freedom sensor, able to measure the absolute orientation, angular and linear acceleration, gravity vector and magnetic field strength. The Razor IMU, connects directly to the Jetson TX2 via USB with the help of an FDTI breakout board. Apart from the aforementioned configuration several other smaller electronic systems are used to support the functionalities of the custom vehicle.

E. Software

A key issue for the adaption of the proposed system in the education and research process, apart from the actual capabilities of the vehicle, is the ease of use and the time needed to develop and test new approaches for autonomous driving. Based on that, we have developed a modular approach which minimizes the development time and allows the user to speed up the development process.

A series of software modules responsible for the control of the devices onboard the vehicle have been developed using

C++ . These modules are implemented onboard the NVIDIA Jetson TX2 and there are also responsible for data logging for post processing analysis.

The devices on board the vehicle have to communicate with the main computer. Some of them, such as the LiDAR have their own SDK that handles connectivity via a serial library and have predefined functions for use. Regarding GPS, IMU and Ultrasonic Sensors, similar architecture modules have been developed using a C++ serial library (LibSerial) for sending command and receiving data. For easy of wiring the serial communication has been established via the USB interface. Respectively software libraries for handling the GPIOs of the Jetson have been used responsible for selecting direction (in/out) and logic level(0/1) of the pins for controlling the stepper drivers.

The external code development is performed using Matlab and by adopting the GPU Coder we have the ability to generate optimized CUDA code from MATLAB code for deep learning, embedded vision, and autonomous navigation. The generated code is portable across NVIDIA GPUs and can be compiled and executed on an NVIDIA Jetson platform. This approach is user friendly, and most of the students attending a related engineering curriculum are familiar with using it, therefore make the platform easily adaptable for educational purposes. The data acquired from experimentation can be easily extracted and used for post processing analysis and further development. The proposed approach is depicted in Fig. 3.

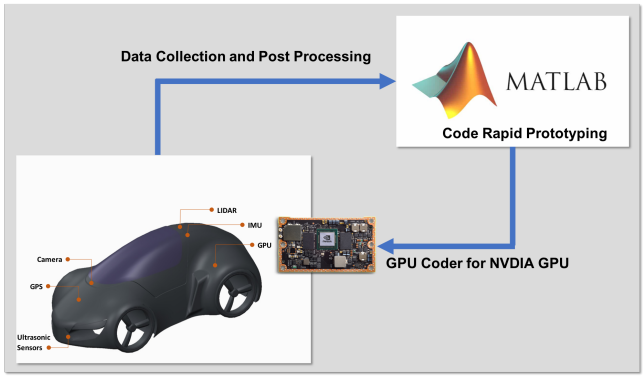


Fig. 3. Software development circle

III. Results - Proof of Concept

To validate our approach several tests have been conducted. In this section we will present two test cases that highlight the functionality of the proposed system. Initially we will demonstrate the ability of the vehicle to move in a predefined path based on the readings from the GPS sensor. This test case emulates, the operation of a real autonomous vehicle in a urban environment which has to operate and navigate in an unknown area using the GPS coordinates. The second test case will demonstrate the ability of the proposed platform to perceive the environment, using the available sensors, identify an obstacle (in our case a pedestrian) and safely avoid it.

To assure that our vehicle is capable of following the desired waypoints we have used a simplification of an ackerman steered vehicle commonly known as the bicycle model. For path tracking we have used the pure pursuit method as it is described in detail in [16]. This approach although simplified provides reliable results that have been verified with extensive experimentation.

It has to be emphasised, that the following experiments have a dual goal. To prove the functionality of the proposed approach and also highlight its robustness which is essential for using it as an educational tool. Therefore our goal wasn't to demonstrate novel approaches in autonomous driving, but adopt well established techniques to prove its functionality.

A. Test Case 1

In the first test case we present the ability of the vehicle to follow a predefined path autonomously without human intervention. Initially we prerecorded a path taking GPS readings and we used this measurements so that we can validate the autonomous operation based on pure pursuit method. Sample trajectories that the vehicle has followed inside the Technical University of Crete's Campus are depicted in Fig. 4 and Fig. 5.

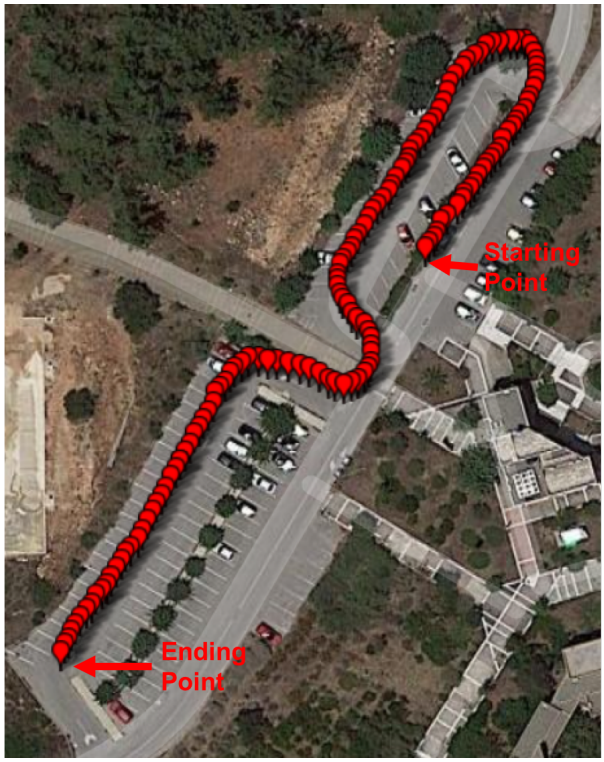


Fig. 4. Vehicle's trajectory inside the TUC campus (Test 1)

B. Test Case 2

In the second test case we present the ability of the vehicle to perceive the environment and avoid a moving obstacle,

Fig. 5. Vehicle's trajectory inside the TUC campus (Test 2)

Fig. 6. Pedestrian detection using stereo camera

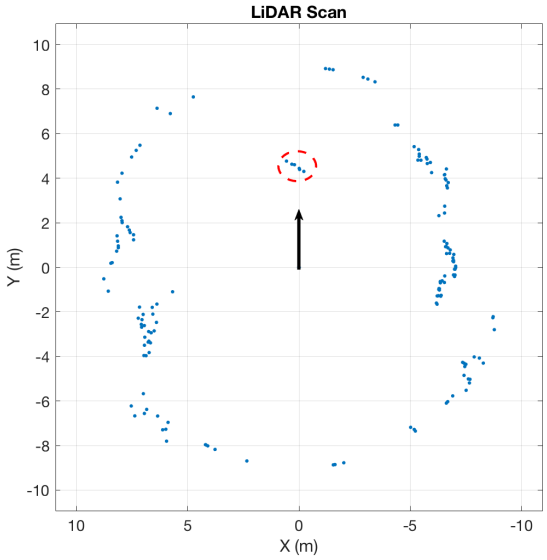


Fig. 7. Data acquired from the lidar sensor

in our case a pedestrian. For this process the vehicle is using the stereo camera and the lidar sensor. For pedestrian detection we have used the pedestrian detection network provided by Matlab. With the help of the GPUCoder, the above network is converted to CUDA code for direct and fast use in the embedded vehicle computer. The results of the identification process are presented in Fig. 6. These are combined heuristically with lidar measurements and as soon as the vehicle detects an obstacle (pedestrian) in a distance smaller that a certain threshold the vehicle stops its operation until the path is clear. A shapshot of the lidar readings is depicted in Fig. 7, where the arrow represents the actual heading of the vehicle, while the readings inside the red eclipse correspond to the pedestrian detected in Fig. 6. This approach also simplistic demonstrates the ability of the testing platform to handle unexpected events as they occur during the autonomous operation of the vehicle.

IV. Discussion and Conclusions

The proposed testbed aims to promote the educational procedure on the domain of autonomous vehicles. Having that in mind, we are planning to provide public access to the software in its final form. In addition, electronic schematics, mechanical blueprints and detailed bill of materials, to replicate the platform, will also be available. Our long term vision is to provide a robust open platform for education and research purposes to the community. We are also planning to fully integrate Robotics Operating System (ROS) in our platform, since currently it can be supported with slight modifications (Jetson TX2 is ROS enabled), and all the sensors selected are ROS compatible. The approach, which we currently followed, was not to install a middle-ware for simplicity and ease of prototyping.

An update version of the proposed system is under development, were we are also considering the need for additional computational power so that we can implement more complicated strategies. The Jetson TX2 has proven robust enough for our purposes. But, in order to keep up with the increasing demands of an autonomous vehicle navigating in a dense environment performing complex functions, other options can also be integrated. For that we have already considered the Drive PX2 which apart from the additional power can also accommodate the already developed CUDA code with minor tweaks and modifications. As far as it concerns perception we are planning on upgrading from the 2D lidar to a 3D capable of scanning at greater distances and providing a 3D representation of the real world.

Ultimately, we are planning to adopt the aforementioned

platform in the educational process (as an experimental testbed in undergraduate and graduate courses) and asses its functionality and usability based on the students' feedback.

ACKNOWLEDGMENTS

This work was partially supported by the the SUNNY Project funded by the European Commission under the FP7 programme Grant Agreement number: 313243.

The authors would like to thank the members of the TUCer team of the Technical University of Crete for their continuous support during the development and experimentation with the autonomous platform.

REFERENCES

[1] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," SAE, Tech. Rep., 2018.

[2] M. Wrner, F. Schuster, F. Dlitzscher, C. G. Keller, M. Haueis, and K. Dietmayer, "Integrity for autonomous driving: A survey," in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, April 2016, pp. 666–671.

[3] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, "Autonomous vehicle technology: A guide for policymakers," RAND Corporation, Tech. Rep., 2016.

[4] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," Victoria Transport Policy Institute, Tech. Rep., 2018.

[5] J. Tani, L. Paull, M. T. Zuber, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: An innovative way to teach autonomy," in *Educational Robotics in the Makers Era*, D. Alimisis, M. Moro, and E. Menegatti, Eds. Cham: Springer International Publishing, 2017, pp. 104–121.

[6] "Donkey car." [Online]. Available: http://www.donkeycar.com/

[7] "Mit racecar." [Online]. Available: https://mit-racecar.github.io/

[8] "F1/10." [Online]. Available: http://f1tenth.org/

[9] "Barc - berkeley autonomous race car." [Online]. Available: http://www.barc-project.com/

[10] "Shell eco-marathon europe." [Online]. Available: https://www.shell.com/energy-and-innovation/shell-ecomarathon/europe.html

[11] "Tucer team." [Online]. Available: http://www.tucer.tuc.gr/el/archiki/

[12] P. Spanoudakis, N. C. Tsourveloudis, G. Koumartzakis, A. Krahtoudis, T. Karpouzis, and I. Tsinaris, "Evaluation of a 2-speed transmission on electric vehicle's energy consumption," in *2014 IEEE International Electric Vehicle Conference (IEVC)*, Dec 2014, pp. 1–6.

[13] P. Spanoudakis and N. C. Tsourveloudis, "On the efficiency of a prototype continuous variable transmission system," in *21st Mediterranean Conference on Control and Automation*, June 2013, pp. 290–295.

[14] D. S. Efstathiou, A. K. Petrou, P. Spanoudakis, N. C. Tsourveloudis, and K. P. Valavanis, "Recent advances on the energy management of a hybrid electric vehicle," in *20th Mediterranean Conference on Control Automation (MED)*, July 2012, pp. 896–901.

[15] A. K. Petrou, D. S. Efstathiou, and N. C. Tsourveloudis, "Modeling and control of the energy consumption of a prototype urban vehicle," in *19th Mediterranean Conference on Control Automation (MED)*, June 2011, pp. 44–48.

[16] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-09-08, 2009.

CoMapping: Efficient 3D-Map Sharing Methodology for Decentralized cases

Luis F. Contreras-Samamé, Salvador Domínguez-Quijada, Olivier Kermorgant and Philippe Martinet*
Laboratoire des Sciences du Numérique de Nantes (LS2N) - École Centrale de Nantes (ECN), 44300 Nantes, France
*INRIA Sophia Antipolis, 06902 Sophia Antipolis, France
Emails: *Luis.Contreras@ls2n.fr, Salvador.Dominguezquijada@ls2n.fr, Olivier.Kermorgant@ls2n.fr, Philippe.Martinet@inria.fr*

***Abstract*—CoMapping is a framework to efficient manage, share, and merge 3D map data between mobile robots. The main objective of this framework is to implement a Collaborative Mapping for outdoor environments where it can not use all the time GPS data. The framework structure is based on 2 stages. The first one, the Pre-Local Mapping Stage, each robot constructs in real-time a pre-local map of its environment using Laser Rangefinder data and low cost GPS information only in certain situations. Afterwards, in the Local Mapping Stage, the robots share their pre-local maps and merge them in a decentralized way in order to improve their new maps, renamed now as local maps. An experimental study for the case of decentralized cooperative 3D mapping is presented, where tests were conducted using 3 intelligent cars equipped with lidars and GPS receiver devices in urban outdoor scenarios. We also discuss the performance of all the cooperative system in terms of map alignments.**

I. Introduction

Mapping challenge can be complex since in certain situations, e.g. for scenarios of large regions, it can require the usage of a group of robots that build the maps in a reasonable amount of time considering accuracy in the map construction [1]. So, a set of robots extends the capability of a single robot by merging measurements from group members, providing each robot with information beyond their individual sensors range. This allows a better usage of resources and executes tasks which are not feasible by a single robot. Multi-robot mapping is considered as a centralized approach when it requires all the data to be analysed and merged at a single computation unit. Otherwise, in a decentralized approach, each robot builds their local maps independent of one another and merge their maps upon rendezvous.

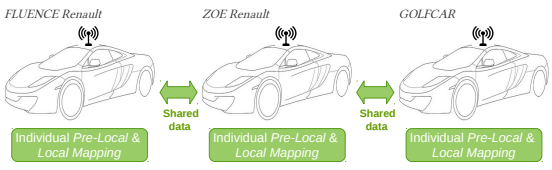


Fig. 1. Scheme of our CoMapping System considering a decentralized case

Figure 1 depicts the scheme of work proposed in this article for a group of robots where it was assumed that ZOE robot have direct exchange of data (as pose, size and limits of maps) with FLUENCE and GOLFCAR. And by contrast, FLUENCE and GOLFCAR are in a scenario of non-direct communication, that is possible in cases where the robots have limited access conditions to a same environment, avoiding to define a meeting point for map sharing between these mobile units.

Following this scenario, this paper presents the development and validation of a new Cooperative Mapping framework (CoMapping) where:

- In the first stage named "Pre-Local Mapping", each individual robot builds its map by processing range measurements from a 3D lidar moving in six degrees of freedom (6-DOF) and using low cost GPS data (GPS/GGA).
- For the second stage named "Local Mapping", the robots send a certain part of their pre-local maps to the other robots based on our proposed Sharing algorithm. The registration process includes an intersecting technique of maps to accelerate processing

This decentralized system is deployed in an outdoor environment without continuous GPS service. Our proposal has been tested and validated in realistic situations. Results include maps developed with data acquired on the surroundings of the ECN (École Centrale Nantes) campus.

II. Related works

In a scenario of cooperative mapping, robots first operate independently to generate individual maps. Here the registration method plays a fundamental role. Many registration applications use Lidar as a Rangefinder sensor for construction of maps [2]. However, a high lidar scan rate compared to its tracking can be harmful for this task, since it is possible the apparition of distortion in the map construction. For those cases, ICP [3] can be applied to match different scans. 2D and 3D lidar implementations with geometric structures matches of a generated local point group were presented in [4] [5]. Those methods use batch processing to build maps with accuracy and hence are not applicable to real-time map construction. In the first stage of our implementation we reconstruct maps as 3D pointclouds in real-time using 3-axis lidar by extraction and matching of geometric features in Cartesian space based in [6] initially. Then our system uses GPS position data to localize that cloud in a global frame.

Once all the maps have been placed in a global frame, they have to be merged together to form a global map. In this context, in [7] proposed a method for 3D merging of

occupancy grid maps based on octrees [8] for multi-robots. Simulation results were presented using Gazebo tool. Maps generated by each simulated robot are stored in files and finally merged offline. For the merging step, an accurate transformation between maps was assumed as known, nevertheless in real applications, that information (the transformation) is not accurate, since in many cases it is obtained by means of uncertain sensor observations that may not offer a reliably information. Contrary, we preformed real experiments for a multi-robot application without supposed known the map transformation. Later, in [9] using an technique pre-merging, which consists in extract from of each map the subset of points included in the common region between maps bounding. Then, a centralized merging process refines the transformation estimate between maps by ICP registration [3] We use a variation of that method [9] but previously we include a efficient technique to exchange maps between robots in order to optimize bandwidth resources of multi-robot network.

On the other hand, other different solutions can be used in order to merge maps for a group of robots. For instance, cases with centralized approach, where the merging is computed on a unit or processing center once the entire environment has been explored by the vehicles, as is presented in [10], [9]. The other approach is the decentralized option, where map merging is executed in different units while traversing the environment, in which this approach considers a meeting point for the vehicles in order to exchange their maps and other data [11], [1], [12]. This last approach is experimentally studied in this paper.

III. Methodology

A. Pre-Local Mapping Stage

Each mobile robot executes a Pre-Local Mapping system using data provided by a LidarSLAM node. We just use GPS position to project the generated map on a global frame, in order to reduce project implementation costs, a beneficial cheap GPS service was used, specifically GPS/GGA(*Global Positioning System Fix Data*) at an accuracy of about 2 to 7 meters. Another advantage of our Pre-Local Mapping Stage is its versatile configuration, since it is not depend on a specific LidarSLAM method. A modified version of the LOAM technique [1] [6] was chosen as LidarSLAM method for this article because it currently ranks first in the KITTI evaluation table [2].

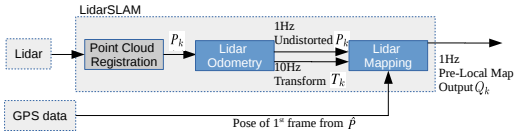


Fig. 2. Architecture of Pre-Local Mapping Stage

Figure 2 illustrates the block diagram of this stage, where $\hat{P}$ is the raw point cloud data generated by a laser scan in the beginning. For each sweep, $\hat{P}$ is registered in the lidar coordinates $\{L\}$. The combined point cloud during each sweep k generates P_k. This P_k is processed by an algorithm named *Lidar Odometry*, which runs at a frequency around 10Hz and receives this point cloud and computes the lidar motion (transform T_k) between two consecutive sweeps. The distortion in P_k is corrected using the estimated lidar motion. The resulting undistorted P_k is processed at a frequency of 1Hz by an algorithm knows as *Lidar Mapping*, which performs the matching and registration of the undistorted cloud onto a map. At last, using the GPS information of the vehicle pose during previous algorithm, it is possible to coarsely project the map of each robot into common coordinate frame for all the robots. This projected cloud is denoted as the Pre-Local Map.

1) Lidar Odometry step: The step begins with feature points extraction from the cloud P_k. The feature points are selected for sharp edges and planar surface patches. Let us define $\overline{S}$ as the set of consecutive points i returned by the laser scanner in the same scan, where $i \in P_k$. A parameter proposed in [6] evaluates the smoothness of the local surface as following,

$$\overline{c} = \frac{1}{\mid \overline{S} \mid . \parallel X^L_{(k,i)} \parallel} \parallel \sum_{j \in S, j \neq i} (X^L_{(k,i)} - X^L_{(k,j)}) \parallel, \qquad (1)$$

where $X^L_{(k,i)}$ and $X^L_{(k,j)}$ are the coordinates of two points from the set $\overline{S}$.

Moreover, a scan is split into four subregions to uniformly distribute the selected feature points within the environment. In each subregion is determined maximally 2 edge points and 4 planar points. The criteria to select the feature points as edge points is related to maximum $\overline{c}$ values, and by contrast the planar points selection to minimum $\overline{c}$ values. When a point is selected, it is thus mandatory that none of its surrounding points are already selected. Other conditions are: selected points on a surface patch can not be approximately parallel to the laser beam, or on boundary of an occluded region.

When the correspondences of the feature points are found based on the method proposed in [6], the distances from a feature point to its correspondence are calculated. Those distances are named as d_E and d_H for edge points and planar points respectively. The minimization of the overall distances of the feature points will allow to obtain the lidar odometry. That motion estimation is modelled with constant angular and linear velocities during a sweep.

Let us define E_{k+1} and H_{k+1} as the sets of edge points and planar points extracted from P_{k+1}, for a sweep $k+1$. The lidar motion relies on establishing a geometric relationship between an edge point in E_{k+1} and the corresponding edge line:

$$f_E(X^L_{(k+1,i)}, T^L_{k+1}) = d_E, i \in E_{k+1}, \qquad (2)$$

where T^L_{k+1} is the lidar pose transform between the starting time of sweep $k+1$ and the current time t_i. T^L_{k+1} contains data about the sensor rigid motion in 6-DOF, $T^L_{k+1} =$

[1]LOAM: https://github.com/laboshinl/loam_velodyne
[2]KITTI ranking: http://www.cvlibs.net/datasets/kitti/eval_odometry.php

$[t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]^T$, wherein t_x, t_y, and t_z are translations along the axes x, y, and z from $\{L\}$, respectively, and θ_x, θ_y, and θ_z are rotation angles, following the right-hand rule.

Similarly, the relationship between an planar point in H_{k+1} and the corresponding planar patch is:

$$f_H(X^L_{(k+1,i)}, T^L_{k+1}) = d_H, i \in H_{k+1}, \tag{3}$$

Equations (2) and (3) can be reduced to a general case for each feature point in E_{k+1} and H_{k+1}, obtaining a nonlinear function, as:

$$\boldsymbol{f}(T^L_{k+1}) = \boldsymbol{d}, \tag{4}$$

in which each row of $\boldsymbol{f}$ is related to a feature point, and $\boldsymbol{d}$ possesses the corresponding distances. Levenberg-Marquardt method [13] is used to solve the Equation (4). Jacobian matrix ($\mathbf{J}$) of $\boldsymbol{f}$ with respect to T^L_{k+1} is computed. Then, the minimization of $\boldsymbol{d}$ through nonlinear iterations allows to solve the sensor motion estimation,

$$T^L_{k+1} \longleftarrow T^L_{k+1} - (\mathbf{J}^T\mathbf{J} + \lambda \mathtt{diag}(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\boldsymbol{d}, \tag{5}$$

where λ is the Levenberg-Marquardt gain.

Finally, the *Lidar Odometry* algorithm produces a pose transform T^L_{k+1} that contains the lidar tracking during the sweep between $[t_{k+1}\ ,\ t_{k+2}]$ and simultaneously an undistorted point cloud $\bar{P}_{k+1}$. Both outputs will be used by the Lidar Mapping step, explained in the next section.

2) Lidar Mapping step: This algorithm is used only once per sweep and runs at a lower frequency (1 Hz) than the Lidar Odometry step (10 Hz). The technique matches, registers and projects the cloud $\bar{P}_{k+1}$ provided by previous step (Lidar Odometry) as a map into the own coordinates system of a vehicle, defined as $\{V\}$. To understand the technique behaviour, let us defined Q_k as the point cloud accumulated until sweep k, and T^V_k as the sensor pose on the map at the end of sweep k, t_{k+1}. The algorithm extends T^V_k for one sweep from t_{k+1} to t_{k+2}, to get T^V_{k+1}, and projects $\bar{P}_{k+1}$ on the robot coordinates system $\{V\}$, denoted as $\bar{Q}_{k+1}$. Then, by optimizing the lidar pose T^V_{k+1}, the matching of $\bar{Q}_{k+1}$ with Q_k is obtained.

In this step the feature points extraction and the finding feature points correspondences are calculated in the same way as in previous step (Lidar odometry), the difference just lies in that all points in $\bar{Q}_{k+1}$ share the time stamp, t_{k+2}.

In that context, nonlinear optimization is solved also by the Levenberg-Marquardt method [13], registering $\bar{Q}_{k+1}$ on the a new accumulated cloud map. To get a points uniform distribution, down-sampling process is performed to the cloud using a voxel grid filter [14] with a voxel size of 5 cm cubes.

Finally, since we have to work with multiple robots, we use a common coordinates system for their maps, $\{W\}$, coming from rough GPS position estimation of the 1st accumulated cloud frame Q_k.

B. Local Mapping Stage

In this section the Local Mapping is detailed, considering that the process is executed on the robot "i" with a shared map by robot "n" (see Figure 3).

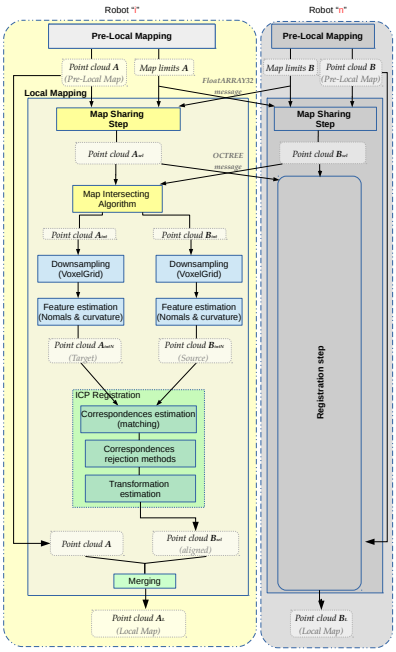


Fig. 3. Architecture of Local Mapping Stage for one robot "i", receiving map data from another robot "n".

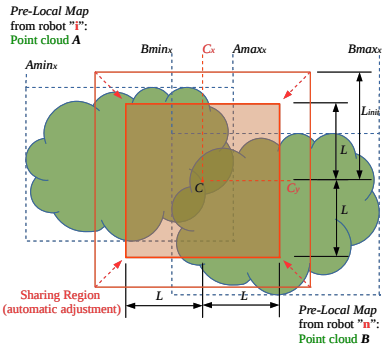


Fig. 4. Graphical representation of the Map Sharing technique (Top view of plane XY). $Amin_x$, $Amax_x$, $Bmin_x$ and $Bmax_x$ represent the point cloud limits along the x-axis.

1) Map Sharing Step: When the generation of Pre-Local Maps is done, the robots would have to exchange their maps to start the maps alignment process. In several cases the sharing and processing of maps of large dimensions can affect negatively the performance of the system with respect to runtime and memory usage. A sharing technique is presented in order to overcome this problem, in which each vehicle builds only sends a certain part of its map to the other robots. When the maps are ready for transferring, they are compressed in octree format using OctoMap library [8] in order to optimize the robot-communication.

The proposed sharing technique is based on the method developed in [15]. Figure 4 depicts the behaviour, wherein point clouds *A* and *B* represent the Pre-Local Maps from two robots "i" and "n" respectively. In each robot the algorithm first receives only information about the 3D limits of the maps (i.e. bounding cubic lattice of the point clouds) and then decides what part of its map will be shared to the other robot. These limits were determined previously using the function $GetBounds()$ that returns two vectors: in the first one $Amin$, their components represent the lowest displacement from the origin along each axis in the point cloud; and the other vector $Amax$ is related to the point of the highest displacement.

Data: Point Cloud A; Limits: Vectors $Amin$, $Amax$, $Bmin$ and $Bmax$; Parameters: Scalars L_{init}, L_{step}, Np_{max}
Result: Point Cloud A_{sel}
begin
  $A_{sel} \leftarrow \emptyset$;
  $C_x = 0$; $C_y = 0$; $C_z = 0$;
  $(V2_x, V3_x) = GetValues(Amin_x, Amax_x, Bmin_x, Bmax_x)$;
  $(V2_y, V3_y) = GetValues(Amin_y, Amax_y, Bmin_y, Bmax_y)$;
  $(V2_z, V3_z) = GetValues(Amin_z, Amax_z, Bmin_z, Bmax_z)$;
  $C_x = (V2_x + V3_x)/2$;
  $C_y = (V2_y + V3_y)/2$;
  $C_z = (V2_z + V3_z)/2$;
  $Np = PointSize(A)$;
  for (L=L_{init} ; $Np > Np_{max}$; $L = L - L_{step}$) **do**
    $Smin_x = C_x - L$; $Smax_x = C_x + L$;
    $Smin_y = C_y - L$; $Smax_y = C_y + L$;
    $Smin_z = C_z - L$; $Smax_z = C_z + L$;
    foreach $\boldsymbol{a} \in A$ **do** ;
    if $Smin_x < a_x < Smax_x$ *and* $Smin_y < a_y < Smax_y$ *and* $Smin_z < a_z < Smax_z$ **then**
      $A_{sel} = A_{sel} + \boldsymbol{a}$;
    end
    $Np = PointSize(A_{sel})$;
  end
end

Algorithm 1: Selection of Point Cloud to share with another robot.

Pseudo-code of the map sharing step is described in Algorithm 1. Inside the code, the function $GetValues()$ sorts in ascending order the array of components along each axis of the vectors $Amin$, $Amax$, $Bmin$, $Bmax$ and returns the 2nd and 3rd values from this sorted array, denoted ($V2$) and ($V3$) respectively. Next, for each axis, the average of the two values obtained by the function $GetValues()$ is used in order to determine the Cartesian coordinates (C_x,C_y,C_z) of the geometric center of the sharing region (S). Actually, this map sharing region is a cube whose edge length $2L$ is determined iteratively. Points from A contained in this cube region are extracted to generate a new point cloud A_{sel}. In each iteration the cube region is reduced until the number of points from A_{sel} is smaller than the manual parameter Np_{max}, which represents the number of points maximum that the user wants to exchange between robots. Once the loop ends, A_{sel} is sent to the other robot. Similarly on the other robotic platform "n", the points from B included in this region are also extracted to obtain and share B_{sel} with the another robot "i". Then, it is worth to remind, the clouds A_{sel} and B_{sel} are encoded and sent in octree format to reduce the usage of bandwidth resources of the multi-robot network. Then maps are decoded and reconverted in 3D point cloud format to be used in the next Registration step. Pointcloud-octree encoding and decoding were realized using ROS nodes supported on OctoMap library [8].

2) Registration Step: The intersecting volumes of the two maps A_{sel} and B_{sel} are computed and denoted as A_{int} and B_{int}, obtained from the exchanged map bounds [9]. In order to improve the computation speed, point clouds A_{int} to B_{int} first go through a down-sampling process to reduce the number of points in the alignment of our clouds. Feature descriptors as surface normals and curvature are used to improve the matching, which is the most expensive stage of the registration algorithm [16]. These generated normal-point clouds A_{intN} and B_{intN} are then used by Iterative Closest Point (ICP) algorithm [17]. This method refines an initial alignment between clouds, which basically consists in estimating the best transformation to align a source cloud B_{intN} to a target cloud A_{intN} by iterative minimization of an error metric function. At each iteration, the algorithm determines the corresponding pairs (***b'***, ***a'***), which are the points from A_{intN} and B_{intN} respectively, with the least Euclidean distance.

Then, least squares registration is computed and the mean squared distance E is minimized with regards to estimated translation $\boldsymbol{t}$ and rotation R:

$$E(R,\boldsymbol{t}) = \frac{1}{Np_{\boldsymbol{b'}}} \sum_{i=1}^{Np_{\boldsymbol{b'}}} \| \boldsymbol{a'}_i - (R\,\boldsymbol{b'}_i + \boldsymbol{t}) \|^2, \qquad (6)$$

where $Np_{b'}$ is the number of points ***b'***.

The resulting rotation matrix and translation vector can be express in a homogeneous coordinates representation (4×4 transformation matrix T_j) and are applied to B_{intN}. The algorithm then re-computes matches between points from A_{intN} and B_{intN}, until the variation of mean square error between iterations is less than an defined threshold. The final ICP refinement for n iterations can be obtained by multiplying the individual transformations: $T_{ICP} = \prod_{j=1}^{n} T_j$. Finally the transformation T_{ICP} is applied to the point cloud B_{sel} to align and merge with the original point cloud A, generating the Local Map A_L then. Each robot thus performed its own merging according to limited data shared from other agents within communication range.

IV. Results

In this section we show results validating the presented concepts and the functionality of our system. As we consider ground vehicles, the ENU (East-North-Up) coordinate system is used as external reference of the world frame $\{W\}$, where

Fig. 5. Vehicles used in the tests: ZOE, FLUENCE and GOLFCAR.

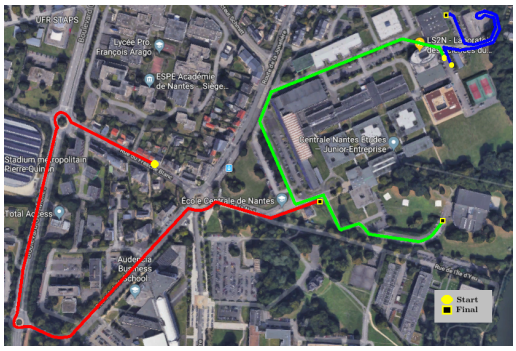


Fig. 6. Paths followed by ZOE (green one), FLUENCE (red one) and GOLFCAR robot (blue one) during experiments. Image source: Google Earth.

y-axis corresponds to North and x-axis corresponds to East, but coinciding its origin with the GPS coordinates [Longitude: -1.547963; Latitude: 47.250229].

In this article, our proposed framework was validated considering three vehicles for experiments, a *ZOE Renault*, a *FLUENCE Renault* and a *GOLFCAR* (see Figure 5) customized and equipped with a *Velodyne* VLP-16 3D lidar, with 360° horizontal field of view and a 30° vertical field of view. All data come from the campus outdoor environment in an area of approximately 1000m x 700m. The vehicles traversed that environment following different paths and collected sensor observations about the world, running pre-local mapping process in real-time.

For the validation, the vehicles build clouds from different paths (see Figure 6). Results of the Pre-Local Mapping of this experiment are shown in Figure 7.

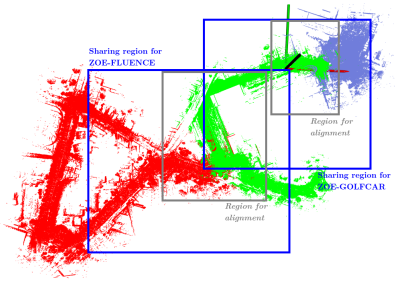


Fig. 7. Top view of unaligned Pre-Local Maps generated by ZOE (green one), FLUENCE (red one) and GOLFCAR robot (blue one) projected on common coordinate system

Figure 7 also depicts the "sharing region" determined during the map exchange process in each robot. It was assumed that all the vehicles have the constraint of exchanging the number of points maximum Np_{max} of 410000 to simulate restrictions in resources of bandwidth network or memory usage in robots. The tests were divided in two. In the first one, test A, ZOE and FLUENCE car define a meeting point to transfer their maps. Once, ZOE car exchanges and updates its local map, a new point of rendezvous for map sharing is determined by ZOE and GOLFCAR in the following test B.

Since we study a decentralized case, then each robot performs a relative registration process considering its Pre-Local map as target cloud for alignment reference. The systems of each robot executes the intersecting algorithm and then an ICP refinement to obtain an improved transform between each map. Figures 8 and 9 depict the intersection between the shared point clouds during the alignment process in each robot. In the yellow box the alignment is more appreciated. Once the refined transformation is obtained, it is then applied to the shared map.

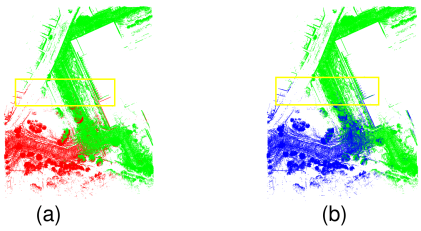


Fig. 8. Test A: Alignment of the intersecting regions with ICP refinement performed in ZOE robot, when it received the FLUENCE map (a) Green and red maps represent the target and source clouds pre ICP, top view (b) Green and blue maps represent the target and aligned source clouds post ICP, top view.

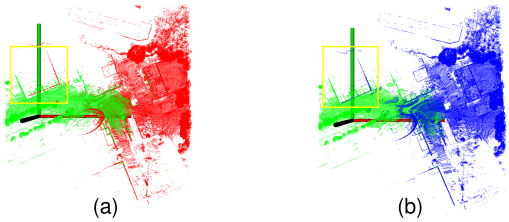


Fig. 9. Test B: Alignment of the intersecting regions with ICP refinement performed in ZOE robot, when it received the GOLFCAR map (a) Green and red maps represent the target and source clouds pre ICP, top view (b) Green and blue maps represent the target and aligned source clouds post ICP, top view.

Quantitative alignment results of the ICP transformation relatives to each robot are shown in Tables I and II. All the ICP transformations are expressed in Euler representation $(x, y, z, roll, pitch, yaw)$ in meters and radians. For instance, first row of Table I corresponds to the merging process in ZOE, when this robot received the map shared by FLUENCE and it aligned that map to its own pre-local map. The decentralized system demonstrated alignments in opposite directions for both robots, since we have to consider that each robot performs the merging process considering its Pre-Local map as target cloud for alignment reference. For instance, on Table II for ZOE vehicle the algorithm converged to the value of displacement of -0.1782 m and -3.2605 m along the x-axis and

y-axis respectively. On the other hand on the GOLFCAR robot, the algorithm converged to a value of displacement of 0.2213 m and 3.3857 m along the x-axis and y-axis respectively, reconfirming relative alignments in opposite directions.

TABLE I
TEST A: RELATIVE ICP TRANSFORMATIONS IN EULER FORMAT BETWEEN ZOE AND FLUENCE ROBOT

Robot	x	y	z	roll	pitch	yaw
ZOE	-1.6517	3.0966	-9.9729	0.0132	0.0730	0.0022
FLU.	4.5748	-4.4556	6.6061	-0.0054	-0.0624	-0.0084

TABLE II
TEST B: RELATIVE ICP TRANSFORMATIONS IN EULER FORMAT BETWEEN ZOE AND GOLFCAR ROBOT

Robot	x	y	z	roll	pitch	yaw
ZOE	-0.1782	-3.2605	1.7771	-0.0516	0.0115	0.0356
GOL.	0.2213	3.3857	-2.6070	0.0411	-0.0256	-0.0380

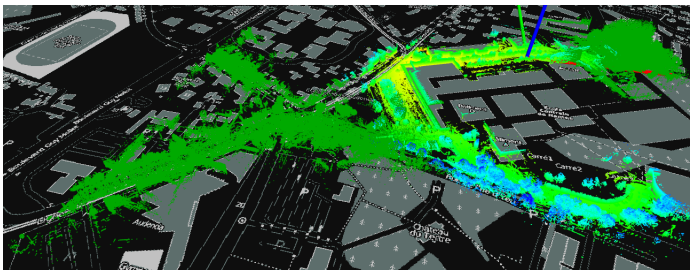


Fig. 10. Final 3D Local Map of ZOE robot

Figure 10 shows one of the merging results corresponding to the ZOE robot, in which the cloud represents the final 3D local map projected on a 2D map in order to make qualitative comparisons. Experiments showed the impact of working with intersecting regions, since it can accelerate the alignment process by decreasing the number of points to compute. In the same way, tests demonstrated that our proposed map sharing technique developed a transcendental position in the performance of the entire mapping collaborative system by reducing the map size to transmit. Finally, the sharing algorithm remains a suitable candidate to exchange efficiently maps between robots considering the use of clouds of large dimensions.

V. CONCLUSION AND FUTURE WORK

A framework, CoMapping, was presented for decentralized 3D mapping system for multiple robots. The work has showed that maps from different robots can be successfully merged, from a coarse initial registration and a suitable exchange of data volume. The system uses initially range measurements from a 3D lidar, generating a pre local maps for each robot. The complete system solves the mapping problem in an efficient and versatile way that can run in computers dedicated to three vehicles for experiments, leading to merged maps independently on each vehicle for GPS-denied environments all the time. Future work will focus on the analysis of maps alignment in decentralized cases, studying the direct impacts on the consistence of maps generated by each robot.

ACKNOWLEDGMENT

This article is based upon work supported by the Erasmus Mundus Action 2 programme through the Sustain-T Project, as well as the institutions ECN (École Centrale de Nantes) and LS2N (Laboratoire des Sciences du Numérique de Nantes). Parts of the equipments used here were funded by the project ROBOTEX, reference ANR-10-EQPX-44-01.

REFERENCES

[1] P. Dinnissen, S. N. Givigi, and H. M. Schwartz, "Map merging of multi-robot SLAM using reinforcement learning." in *SMC*. IEEE, 2012. [Online]. Available: http://dblp.uni-trier.de/db/conf/smc/smc2012.html#DinnissenGS12

[2] S. Kohlbrecher, O. V. Stryk, T. U. Darmstadt, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *in International Symposium on Safety, Security, and Rescue Robotics. IEEE*, 2011.

[3] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets - open-source library and experimental protocol." *Auton. Robots*, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/arobots/arobots34.html#PomerleauCSM13

[4] R. Zlot and M. Bosse, "Efficient Large-Scale 3D Mobile Mapping and Surface Reconstruction of an Underground Mine." in *FSR*, ser. Springer Tracts in Advanced Robotics, K. Yoshida and S. Tadokoro, Eds. Springer, 2012. [Online]. Available: http://dblp.uni-trier.de/db/conf/fsr/fsr2012.html#ZlotB12

[5] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a Spring-Mounted 3-D Range Sensor with Application to Mobile Mapping," *IEEE Transactions on Robotics*, Oct 2012.

[6] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[7] J. Jessup, S. N. Givigi, and A. Beaulieu, "Merging of octree based 3D occupancy grid maps," in *2014 IEEE International Systems Conference Proceedings*, March 2014.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Auton. Robots*, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0

[9] J. Jessup, S. N. Givigi, and A. Beaulieu, "Robust and Efficient Multirobot 3-D Mapping Merging With Octree-Based Occupancy Grids," *IEEE Systems Journal*, 2015.

[10] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, July 2006.

[11] N. E. Özkucur and H. L. Akin, "Supervised feature type selection for topological mapping in indoor environments," in *21st Signal Processing and Communications Applications Conference, SIU 2013, Haspolat, Turkey, April 24-26, 2013*, 2013, pp. 1–4. [Online]. Available: http://dx.doi.org/10.1109/SIU.2013.6531556

[12] J. Zhang and S. Singh, "Aerial and Ground-based Collaborative Mapping: An Experimental Study," in *The 11th Intl. Conf. on Field and Service Robotics (FSR)*, Sep 2017.

[13] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[14] R. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011.

[15] L. Contreras, O. Kermorgant, and P. Martinet, "Efficient Decentralized Collaborative Mapping for Outdoor Environments," in *2018 IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, United States, Jan 2018 - In Press.

[16] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Jun. 2001.

[17] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791

Single-View Place Recognition under Seasonal Changes

Daniel Olid, José M. Fácil and Javier Civera

***Abstract*— Single-view place recognition, that we can define as finding an image that corresponds to the same place as a given query image, is a key capability for autonomous navigation and mapping. Although there has been a considerable amount of research in the topic, the high degree of image variability (with viewpoint, illumination or occlusions for example) makes it a research challenge.**

One of the particular challenges, that we address in this work, is weather variation. Seasonal changes can produce drastic appearance changes, that classic low-level features do not model properly. Our contributions in this paper are twofold. First we pre-process and propose a partition for the Nordland dataset, frequently used for place recognition research without consensus on the partitions. And second, we evaluate several neural network architectures such as pre-trained, siamese and triplet for this problem. Our best results outperform the state of the art of the field. A video showing our results can be found in `https://youtu.be/VrlxsYZoHDM`. The partitioned version of the Nordland dataset at `http://webdiis.unizar.es/~jmfacil/pr-nordland/`.

I. Introduction

Visual place recognition consists on, having a query image, retrieving from a database another image that corresponds to the same place, see Fig. 1. Place recognition plays a relevant role in several applications, *e.g.* mobile robotics. To name a few, place recognition can be used for topological mapping [1], for loop closure and drift removal in geometric mapping [2], and for learning scene dynamics in lifelong localization and mapping [3].

Place recognition for robotics presents multiple challenges. For example, most of the times the places databases are huge and the retrieval time is constrained by the real-time operation of robots. Another relevant challenge, which is the one we will address in this paper, is the variability in the visual appearance of the places. The appearance variations might have different sources: viewpoint and illumination changes, occlusions and scene dynamics.

The appearance changes coming from different viewpoints and illumination conditions, assuming a static scene, have been addressed quite successfully. Local point features (e.g., SIFT, SURF and ORB), based on image gradients, show a high repeatability and descriptor invariance to moderate levels of illumination and viewpoint changes. Checking the geometric and sequential compatibility of such local features can improve even further their robustness [4]. Global image features have been also used for place recognition [5], [6], showing better scalability but lower performance under viewpoint changes or occlusions.

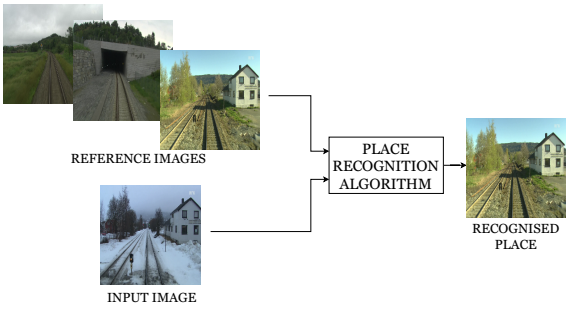


Fig. 1. Place recognition overview. The inputs are two; a database of images taken in different places, and query view imaging the place to recognize. The output is an image of the database showing the place of the query image.

The classical approaches based on hand-designed low-level features are, however, limited for the representation of dynamic scene changes. There has been several works aiming at designing descriptors with higher invariance to certain transformations, either based on models (e.g., [7]) or based on learning from data (e.g., [8]). The most recent approaches use Convolutional Neural Networks (CNNs), due to their higher potential to learn image patterns. In this work we explore the use of CNNs for place recognition in the particular case of seasonal changes. Our specific contributions over the state of the art are:

- We have trained a weather-invariant place recognition method, based on CNNs. We use CNNs to extract image descriptors, that we compare using the Euclidean distance. Fig. 2 depicts some of the weather variations considered.
- We have designed a dataset using images extracted from the Nordland videos [9]. We propose our Nordland dataset partition as a common framework for evaluating place recognition.
- We have compared our results in the Nordland dataset against other state of the art techniques. Our method is capable of correctly recognizing 98% of the input places in 80km routes under favorable conditions and 86% under drastic appearance changes like the ones occurring between summer and winter.

The rest of this paper is structured as follows. Section II analyzes the related work in place recognition. Section III explains the development of the dataset. Section IV introduces the neural network architectures that we have

This work was partially supported by the Spanish government (project DPI2015-67275) and the Aragón regional government (Grupo DGA-T45_17R/FSE

The authors are with the I3A, Universidad de Zaragoza, Spain
`danielolid94@yahoo.es, jmfacil@unizar.es, jcivera@unizar.es`

Fig. 2. Images from the same place in different seasons. From top-left and clockwise: winter, summer, fall and spring. Notice the appearance change due to different weather conditions. The images have been extracted from the videos of the Nordland dataset.

used. Section V presents our results. Finally, in section VI we summarize our conclusions.

II. Related work

The most common approaches to place recognition are based on local image features using classic extractors and descriptors. Two of the most relevant among these techniques are FAB-MAP [10] and DBoW [4]. The performance of these algorithms is excellent for moderate viewpoint and illumination changes, but it decreases for other types of appearance variations.

An alternative approximation consists in using neural networks as feature extractors. Sünderhauf *et al.* analyzed in [11] the use of neural networks for the purpose of place recognition with promising results. [11], [12] and [13] were the first ones to use neural networks for this purpose but [14] and [15] were the first ones to specifically train neural architectures to attack this problem. There is no consensus on what kind of architecture is better for this task.

In this work we compare three different techniques that can be considered state of the art in place recognition: Unsupervised linear learning techniques for visual place recognition [16], deep learning features at scale for visual place recognition [15] and CNN for appearance-invariant place recognition [14].

The first method [16] applied principal components analysis to reduce the dimensionality of the feature vector, eliminating the dimensions that are affected by appearance changes. The second method [14] used a triplet neural network architecture to fine-tune a pre-trained model and improve the robustness of the extracted features. Their network learned to map images to a vector space where euclidean distance represents similarity. The third method [15] trained a deep neural network to classify the place that appeared in a dataset of images taken from surveillance cameras.

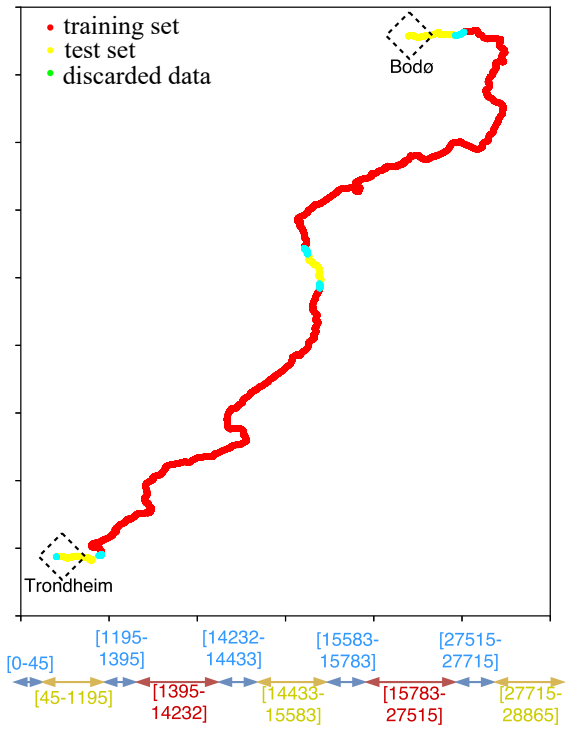


Fig. 3. Proposed dataset partition for the Nordland dataset. **Top:** Geographical representation of the training (red) and test (yellow) sets. **Bottom:** Index representation of the distribution, w.r.t. frame index in the videos.

This work develops a technique similar to the one implemented in [14]. As a novelty, we also train siamese neural networks and consider different pre-trained networks.

III. The Nordland Dataset: Pre-Processing and Partitions

In this work, we have used the Nordland railroad videos. In 2012, the Norway broadcasting company (NRK) made a documentary about the Nordland Railway, a railway line between the cities of Trondheim and Bod. They filmed the 729km journey with a camera in the front part of the train in winter, spring, fall and summer. The length of each video is about 10 hours and each frame is timestamped with the GPS coordinates.

This dataset has been used by other research groups in place recognition, for example [14] and [16]. Each group uses different partitions for training and test, making difficult to reproduce the results. In this work we propose a specific partition of the dataset and a baseline, to guarantee a fair comparison between algorithms. Our intention is to release the processed dataset if the paper is accepted.

A. Data pre-processing

The first step, creating the dataset, was to extract the maximum number of images from each video. Moreover, GPS data corruption was fixed and we also eliminated tunnels and stations. After these steps, grabbing one frame

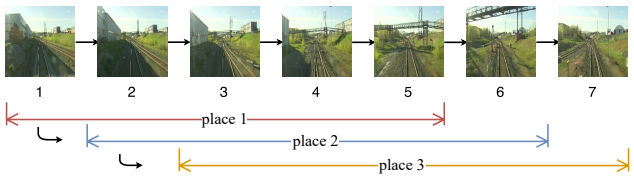


Fig. 4. A sliding window of five images is considered in this work as the same place. Notice the similarity of consecutive images. The figure is best viewed in electronic format.

per second, we obtained $28,865$ images per video. We used speed information from the GPS data to filter stations and a darkness threshold to filter tunnels.

B. Dataset partitions

Fig. 3 illustrates the partition of the whole image set in the Nordland dataset. We decided to create the test set with three different sequences of $1,150$ images (a total of $3,450$, yellow in the figure). The rest of the images were used for training ($24,569$, red in the figure). By using multiple sections, the variety of places and appearance changes contained in the test set increases. We also left a separation of a few kilometers between each test and train section by discarding some images in order to guarantee the difference between test and train data.

C. Place labels

Given the similarity between consecutive images, in this work we propose to consider that two images are of the same place if temporally they are separated by 3 images or less. We applied a sliding window of 5 images over the whole dataset in order to group images taken from five consecutive seconds. This process can be seen in Fig. 4.

IV. Neural Network Architectures

Fig. 5 shows the functional blocks of the proposed place recognition method. Our goal was to train a network to extract feature vectors that are close to the ones extracted from images of the same place, even in the presence of appearance changes. Our similarity metric is the Euclidean distance. We acknowledge that the distance function plays an important role in the feature space distribution, loss and optimization convergence. However, we preferred to focus our efforts on other parts of the problem rather than experimenting with other alternatives, e.g., the cosine distance.

We studied three different ways of using neural networks. First of all, we evaluated the performance of features extracted by pre-trained networks. We then proceeded to train siamese and triplet architectures specifically for the problem of place recgonition.

A. Pre-trained networks

In [11], Sünderhauf *et al.* studied the performance of features from different neural networks for the purpose of place recognition. In this work, we analyzed the features extracted by some layers of the popular VGG-16 model [17], which was trained on Imagenet. Fig. 6 shows the structure

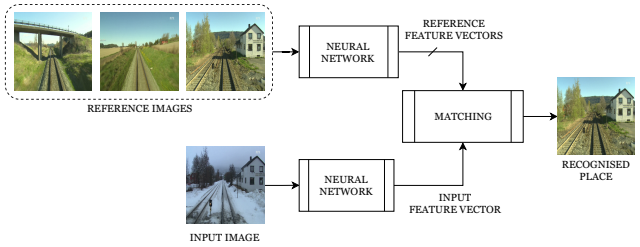


Fig. 5. Overview of the place recognition algorithm. First, we extract a descriptor for every (visited place) image in the database. Second, for every new image (query) we extract its descriptor and compare it with those extracted from the database. The retrieved place will be the one with the most similar descriptor.

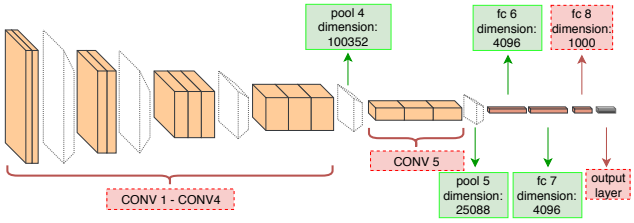


Fig. 6. VGG-16 Layers. *In red:* Layers not used. *In green:* Used layers.

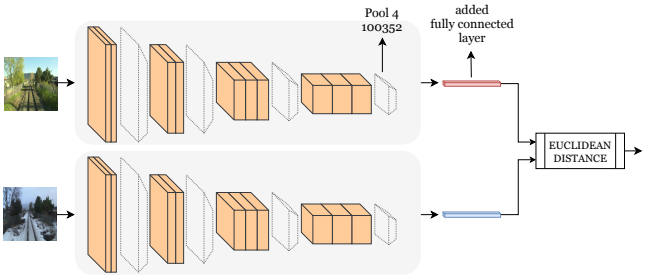


Fig. 7. Siamese architecture used in our work. We show in gray the pre-trained CNN blocks. The fully-connected layer added has 128 neurons.

of the model and the layers that we have evaluated. We have also evaluated the performance of the same architecture trained for scene recognition on the Places dataset.

In the rest of this paper, by extracted feature vector we refer to the output of the neural network at the chosen layer after the non-linear activation. In the case of convolutional layers, we flattened the output tensor.

B. Siamese networks

Siamese neural networks, proposed in [18], are capable of improving the robustness of pre-trained descriptors for place recognition. We modified the VGG-16 model in order to use a siamese architecture and added a new fully-connected layer (without activation function) after the one that showed the best performance in the pre-trained experiments. The final structure is showed in Fig. 7. Training was done for 5 epochs with $834,746$ positive pairs (two images of the same place with different appearance) and $834,746$ negative pairs (two images of different places) taken from the previously mentioned training dataset. We used the contrastive loss [19].

C. Triplet networks

As mentioned in Section II, Gómez-Ojeda *et al.* [14] were the first ones to train triplet networks with this purpose. Triplet neural networks improve the results of siamese architectures by training positive and negative pairs at the same time. Moving closer the descriptors from the same place and apart the descriptors from different places in the same instant leads to a more stable and efficient learning process.

In order to use a triplet architecture, we modified the VGG-16 pre-trained model by adding a new fully-connected layer (without activation function) after the layer that performed better in the pre-trained experiments. We trained the new layer with $834,746$ image triplets for 5 epochs. The loss function used in this case was the *Wohlhart-Lepetit* loss. This loss, proposed in [20] was also used in [14]:

$$E = max\left\{0, 1 - \frac{d_n}{margin + d_p}\right\} \tag{1}$$

Where E is the loss error, d_p is the distance between the positive and neutral input, d_n is the distance between the neutral and negative input and $margin$ is a parameter that limits the difference between the distances.

In this function, the loss is zero when the positive pair is closer than the negative pair plus the margin. Moreover, the loss value is limited between 0 and 1. We set the margin value to 1 in all our experiments.

V. Experimental Results

In order to evaluate our deep models, we used the images from one season as reference and images of a different season as query (summer against winter, winter against fall, etc.). Each image is processed by the neural network to produce the feature vector. After the extraction, each feature vector is compared with every feature vector of every reference season, and the closest one is considered the place predicted by the algorithm. This process is repeated for each one of the $3,450$ test images. The number of times that the closest place is the correct one gives the the fraction of correct matches fc, which is the metric that we have used.

$$\text{fc} = \frac{\text{\# of correct predicted places}}{\text{\# of evaluated places}}, \tag{2}$$

It is important to note that we consider a match is correct when the closest feature vector corresponds to a place within a 5-frames window. The distance between the feature vectors measures the confidence of the result and a distance threshold can be applied to obtain precision-recall curves. We have preferred to focus our analysis on the robustness of the extracted features.

A. Pre-trained

Fig. 8 shows the results obtained from the original VGG-16 pre-trained model. Out of all the studied layers, we found that features extracted from the fourth pooling layer (*pool*4) had the highest fraction of correct matches in all the season combinations. The results are worse as the layers are closer to the VGG-16 output. The main reason might be that, as the dimension of the layer decreases, some of the information that is robust to appearance changes is lost. Moreover, the last layers of the model contain semantic information which is specific to the original problem.

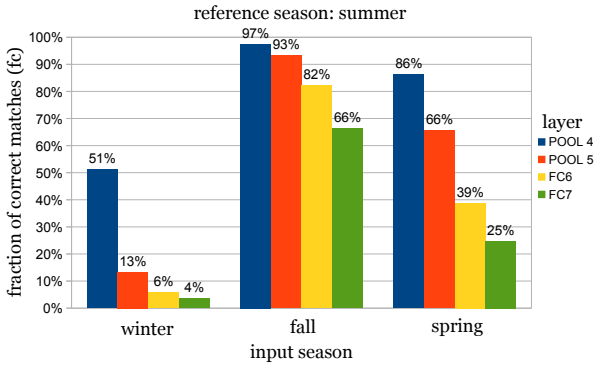


Fig. 8. Fraction of correct matches using the pre-trained VGG-16 layers as feature extractors with summer as reference season and the other seasons as input.

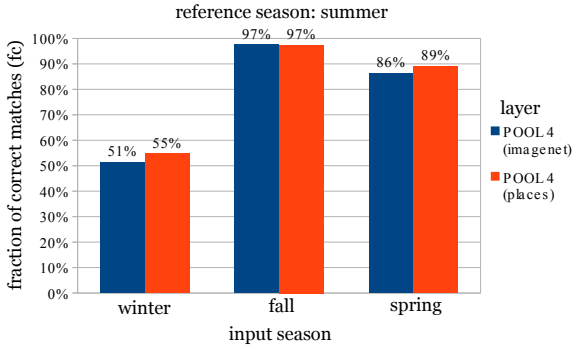


Fig. 9. Fraction of correct matches using features from the *pool4* layer of VGG-16. We compare the Imagenet pre-trained version vs the Places dataset pre-trained one. We show the results with summer as reference season and the other seasons as input.

After that, we compared the performance of the original VGG-16 model to the VGG-16 model trained on the Places dataset by evaluating the features extracted from the *pool*4 layer. We observed that the model trained for scene-recognition achieved better results in all the studied combinations, as shown in Fig. 9. The main reason behind this is that, in order to classify scenes, the internal layers of the model have learned to extract features that are more useful for place recognition.

In the rest of our experiments, we decided to use the fourth pooling layer of the VGG-16 trained on the Places dataset as the starting point. The extracted feature vectors have a dimension of $100,352$.

B. Siamese and triplets

After several experiments, we observed that a descriptor size of 128 is sufficiently discriminative for place recognition. Increasing the size of the layer increases the computational cost without a significant improvement in the accuracy.

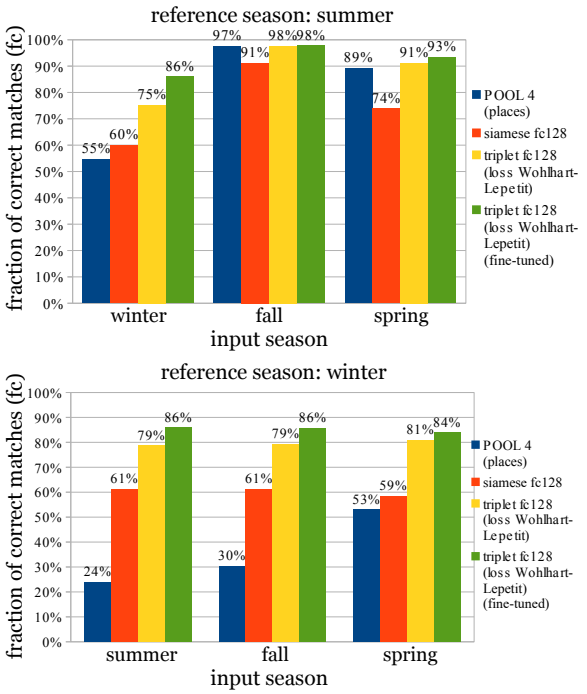


Fig. 10. Fraction of correct matches using different strategies: Pre-trained, siamese and triplet networks (with and without fine-tuning). *Top:* Results with summer as reference season and the other seasons as input. *Bottom:* Results with winter as reference season and the other seasons as input.

Fig. 10 compares the results obtained with the pre-trained, siamese and triplet architectures. The pre-trained network only outperformed the siamese architecture in some combinations where summer images were used as reference. It should be noted that the siamese feature vector has 128 dimensions, while the pre-trained one has $100,352$. Even if the siamese network has not outperformed all the pre-trained results, the siamese architecture has learnt to extract a much smaller feature vector, while keeping the discriminative information.

On the other hand, the triplet network outperformed the siamese and pre-trained models in all the studied combinations. The triplet results that we show in Fig. 10, belong to two different experiments. In our first experiments, we trained the newly added layer (*triplet fc128 - loss Wohlhart Lepetit*). We then proceeded to train the layer while fine-tuning the rest of the VGG-16 pre-trained structure (*triplet fc128 - loss Wohlhart Lepetit - fine-tuned*). It can be observed that the accuracy of the fine-tuned model is higher.

We conclude that the best results were obtained with the fine-tuned triplet network, starting from the weights of the pre-trained VGG-16-Places and adding a fully-connected layer with an output dimension of 128.

Table I shows the fraction of correct matches achieved for every possible combination of reference-input seasons.

C. Comparison against other approaches

Fig. 11 shows the comparison between our results, the PCA technique of [16] and the two neural network models

TABLE I

FRACTION OF CORRECT MATCHES FOR EVERY SEASON COMBINATION.

input \reference	summer	fall	winter	spring
summer	—	0.8548	0.8591	0.9545
fall	0.9777	—	0.8583	0.9562
winter	0.8597	0.9771	—	0.9545
spring	0.9336	0.94	0.8388	—

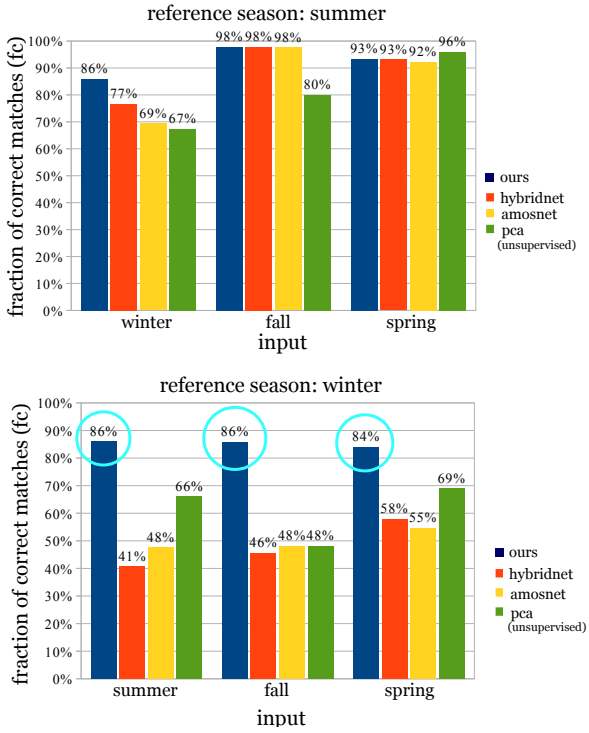


Fig. 11. Fraction of correct matches comparison: our work, Hybridnet, Amosnet and the unsupervised PCA technique. *Top:* Results with summer as reference season and the other seasons as input. *Bottom:* Results with winter as reference season and the other seasons as input.

trained in [15] (Hybridnet and Amosnet). The comparison is made using summer and winter as the reference seasons. Notice that our model matches or outperforms the other techniques in almost every combination, and particularly in those with drastic appearance changes. In the most challenging cases (the ones with winter as reference) the best result is obtained with summer and fall as input seasons, where our model achieved 86% of correct matches while the second best, the unsupervised PCA [16], obtained less than 66%.

The unsupervised PCA results were obtained from their original paper [16]. For Hybridnet and Amosnet, we downloaded the models from the authors of [15] and tested their performance in the test partition of our dataset.

Finally, Fig. 12 shows two examples of correct matches. Notice that our method is robust to strong changes produced by snow and illumination. Fig. 13 shows two examples of incorrect matches. Notice that both are difficult even for a human. The similarities in the geographical features of those places make them look like the same place.

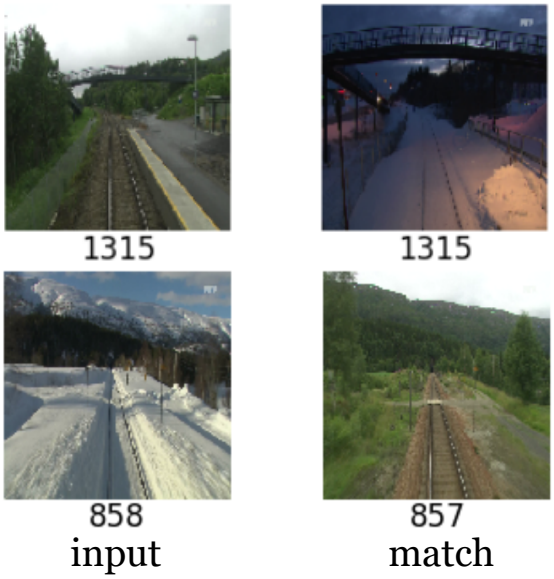


Fig. 12. Places correctly recognized by our algorithm. The index in the sequence is shown at the bottom of each image.

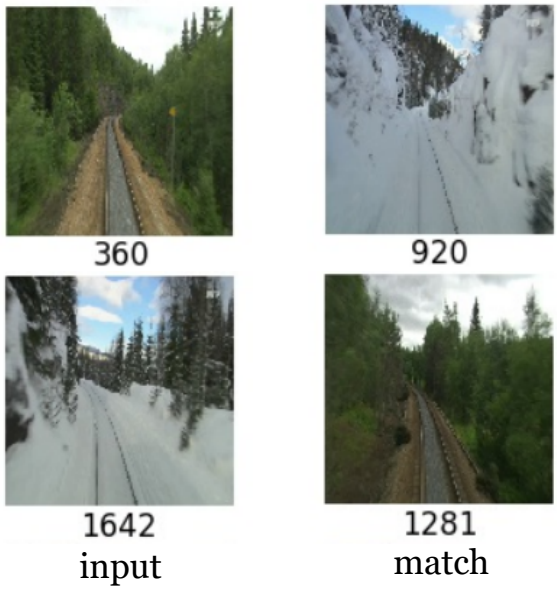


Fig. 13. False positive examples from our algorithm. The index in the sequence is shown at the bottom of each image. Notice that these particular places are difficult even for a human.

VI. Conclusions

In this work we have implemented a place recognition method which is robust to appearance changes, in particular to those caused by weather conditions. Our proposal works by training a neural network to extract a descriptor, that can be compared with others using the Euclidean distance.

Our experiments show that siamese and triplet neural networks learn robust features to appearance changes. Triplet neural networks achieved better results than siamese ones. We show that a VGG-16 model trained on the Places dataset shows a reasonable performance, improved by fine-tuning. Finally, we have shown that our method achieves state-of-the-art results in place recognition on the Nordland dataset.

References

[1] E. Garcia-Fidalgo and A. Ortiz, "Hierarchical place recognition for topological mapping," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1061–1074, 2017.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps-summarizing experience for lifelong localization," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2767–2773, IEEE, 2015.

[4] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[5] N. Sünderhauf and P. Protzel, "Brief-gist-closing the loop by simple means," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1234–1241, IEEE, 2011.

[6] A. C. Murillo, G. Singh, J. Kosecká, and J. J. Guerrero, "Localization in urban environments using a panoramic gist descriptor," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 146–160, 2013.

[7] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "Dali: deformation and light invariant descriptor," *International journal of computer vision*, vol. 115, no. 2, pp. 136–154, 2015.

[8] Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "Tilde: A temporally invariant learned detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5279–5288, 2015.

[9] N. B. Corporation, "Nordlandsbanen: minute by minute, season by season," 2012. [Online; accessed 19-June-2017].

[10] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[11] N. Sünderhauf, F. Dayoub, S. Shirazi, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," *CoRR*, vol. abs/1501.04158, 2015.

[12] P. Neubert and P. Protzel, "Local region detector+ cnn based landmarks for practical place recognition in changing environments," in *Mobile Robots (ECMR), 2015 European Conference on*, pp. 1–6, IEEE, 2015.

[13] A. Gout, Y. Lifchitz, T. Cottencin, Q. Groshens, S. Griffith, J. Fix, and C. Pradalier, *Evaluation of off-the-shelf cnns for the representation of natural scenes with large seasonal variations*. PhD thesis, UMI 2958 GeorgiaTech-CNRS; CentraleSupélec UMI GT-CNRS 2958 Université Paris-Saclay, 2017.

[14] R. Gomez-Ojeda, M. Lopez-Antequera, N. Petkov, and J. Gonzalez-Jimenez, "Training a convolutional neural network for appearance-invariant place recognition," *arXiv preprint arXiv:1505.07428*, 2015.

[15] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep learning features at scale for visual place recognition," *arXiv preprint arXiv:1701.05105*, 2017.

[16] S. Lowry and M. J. Milford, "Supervised and unsupervised linear learning techniques for visual place recognition in changing environments," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 600–613, 2016.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.

[19] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 1735–1742, IEEE, 2006.

[20] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3109–3118, 2015.

Future Depth as Value Signal for Learning Collision Avoidance

Klaas Kelchtermans[1] and Tinne Tuytelaars[1]

Abstract— **The constant miniaturization of robots reduces the array of available sensors. This raises the need for robust algorithms that allow robots to navigate without collision based solely on monocular camera input. Towards this goal, we propose a new learning-based method for the task of obstacle avoidance. We propose a neural network policy for monocular collision avoidance with self-supervision that does not require actual collisions during training. To this end, we demonstrate that a neural network is capable of evaluating an input image and action by predicting the expected depth in the future. In this sense, the future depth can be seen as an action-value-signal. In comparison with our baseline model that is based on predicting collision probabilities, we show that using depth requires less data and leads to more stable training without need for actual collisions. The latter can be especially useful if the autonomous robot is more fragile and not capable to deal with collisions (e.g. aerial robots). The proposed method is evaluated thoroughly in simulation in a ROS-Gazebo-Tensorflow framework and will be made available on publication[2].**

I. INTRODUCTION

Collision avoidance is one of the core tasks of autonomous navigation besides road following and destination pursuing. Smaller robots solely equipped by a light-weight camera and a small GPU are capable of performing more and more complex tasks. General collision avoidance remains however challenging. Methods based on tracking keypoints and keeping a map combined with path-planning have gained impressive results [1]. However, these methods are unreliable in case of blurred images, abrupt motions or lack of features to track. In order to build an algorithm that can deal with new situations and that can adjust its features in a data-driven fashion without having to tweak many parameters, we look at learning algorithms. These systems have the benefit of learning and adapting from their mistakes which makes them more suitable for dynamic environments[2].

Deep neural networks (DNN) have succeeded at increasingly more complex tasks in computer vision and reinforcement learning [3], [4], [5], [6]. Convolutional neural networks (CNN) can handle high dimensional input data, like monocular RGB images, thanks to the parameters shared spatially. Earlier attempts to use CNNs to predict control relied on imitation learning in order to learn models to imitate an expert (human) that collected the data [7]. Other work succeeded at training a CNN to learn a quadcoptor to follow forest trails based on a big dataset collected with a straight-, left- and right-looking camera . The CNN was trained with supervised learning [8] on this set of labeled data. In [9], an iterative procedure (DAGGER) was demonstrated, using a dataset that was partially collected by a human and partially by the policy being trained, resulting in an aggregated dataset. Both methods required a big amount of annotated data which is impractical in most applications. Moreover, using a demonstration flight to represent collision avoidance might not be the best setup as there can be many equally valid routes.

Alternatively, one can decouple feature extraction and control. For instance, in Michel's car [10] a CNN is used to estimate depth upon which a separate reinforcement learning algorithm is applied. Similarly, in [11], estimated depths are used as input for a behavior arbitration algorithm in order to extract a policy for indoor navigation. The behavior arbitration algorithm however needs a number of parameter tweaking steps that differ for each environment and robot.

Recent work by Kahn et al. [12] comes closest to this work. They succeed at making a deep recurrent neural network (RNN) perform monocular autonomous navigation through a lengthy corridor. They introduce a reinforcement learning method, called generative computation graph, that is learned with a self-supervised reward signal. The reward is negative for each collision. This collision is detected by a sudden change of inertia coming from the IMU. As the labeling happens without human intervention we refer to this as self-supervision. The neural network evaluates a number of series of future actions by estimating the probability of a collision within the next horizon of frames.

The main drawback of this system however is the need for collisions. For many robots, this is impossible or at least very expensive, especially if you want to apply it to aerial robots. In this work, we therefore explore an alternative system that uses depth as a self-supervised reward. This avoids the requirement for numerous collisions and might actually learn in a more stable way than predicting sparse collision probabilities. If the goal is to avoid any close obstacle, a depth scan seems to be very indicative of which direction is to be preferred. Following this reasoning one could think of the depth scan as an objective that should be maximized. To simplify the setup in this exploratory phase of the project, we decided to work with a LiDAR as depth sensor rather than an estimated depth map. This suffices as a proof-of-concept. At a later stage this LiDAR could be substituted with a depth estimation network taking monocular camera as input. Although this might seem to be a big assumption, [10] and [11] already indicated the potential of estimated depth maps as well as most recent work [13].

The research question we investigate in this work is whether depth seen at future time-steps can be used as a value-function for evaluating the current state-action-pair for

[1]The authors are with KU Leuven, ESAT-PSI, imec, Belgium. firstname.lastname@esat.kuleuven.be

[2]Project page: kkelchte.github.io/depth-q-net

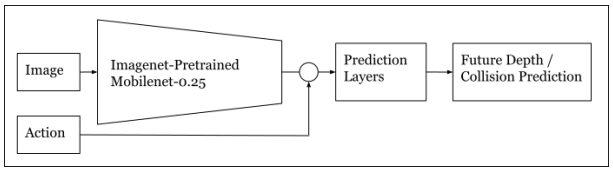


Fig. 1. Shared architecture of collision and depth prediction network.

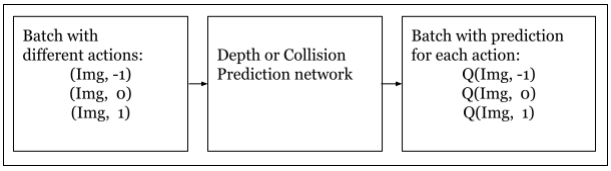


Fig. 2. Evaluation of a batch over different actions in one forward pass.

the application of collision avoidance. In order to demonstrate potential benefit we compare our method with a baseline model that uses real collisions as in [12]. Besides overcoming the need for collisions, we demonstrate how predicting action-dependent future depth leads to more stable learning behavior requiring much less training data.

In the remainder of the paper we first describe the general background (Section II). Section III describes our method in more detail as well as our baseline model. Section IV shows the experimental setup as well as the results. In the appendix the reader can find more details on the training procedure.

II. BACKGROUND

The annotation as well as most of the formulas are based on [2] although simplified for readability. In reinforcement learning, an agent interacts with its environment by performing actions. The agent tries to find a policy π that maps current state s to an action a at time t in order to maximize the expected accumulated discounted reward G, named the return. This is estimated by the state-action value function or Q-value:

$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,T}[\sum_{k=0}^{H} \gamma^k R_{t+k+1} | s_t, a_t]$

in which γ represents the discount factor and R the expected reward. The value-function depends on the one hand on the environment bringing the agent to the next state according to a transition model, $T(s_{t+1}|s_t, a_t)$, and on the other hand the policy $\pi(a_t|s_t)$ picking the action from that next state.

Our algorithm is model-free in the sense that it does not try to model the transition function T explicitly. By definition, the value-function Q should be the sum over infinitely many time steps. However, it is often preferred to work with a finite horizon H. A specific set of algorithms are called myopic, which means that they only care about the immediate reward, $Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,T}[R_{t+1}|s_t, a_t]$. This corresponds to a horizon H of 1.

In deep reinforcement learning (DRL), the value-function is approximated by a neural network. The neural network is then trained on experiences the agent has collected by interacting with its environment: (s_t, a_t, r_t, s_{t+1}). Sampling batches of experiences in order to train the algorithm is called experience replay. The value-function can then be updated using the bellman-equation in a temporal difference setting:

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

In this situation $Q(s_{t+1}, a_{t+1})$ is referred to as the bootstrap. If the experience is collected by the policy being trained, the bootstrap can be estimated for the same policy. In this case the algorithm is called on-policy. In some algorithms the data can be collected by a different policy in which case they are referred to as off-policy. This is very beneficial as it allows collected data to be reused for training multiple agents overcoming excessive amounts of experience collection. Our proposed method can be trained off-policy.

III. METHOD

In this section we first explore the feasibility of using depth as a reward or a value-signal in a collision-avoidance setting. Later we explain the architecture of our Depth-Q-net. In the end, we explain our baseline model based on collision prediction inspired by the work of Kahn [12].

Feasibility of Depth as Value- or Reward-signal

It appears that when using depth as a reward or return function, both lead to a surprising contradiction:

Let's first look at a situation where the difference in depth is given as a **reward**. This means that an increase in depth corresponds to a positive reward. However, mathematically this leads to a value-function, V_t, that is similar to the negative of the depth:

$V_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$

$V_t = D_{t+1} - D_t + \gamma(D_{t+2} - D_{t+1}) + \gamma^2(D_{t+3} - D_{t+2}) + ...$

$V_t = -D_t + (1-\gamma)D_{t+1} + \gamma(1-\gamma)D_{t+2} + ...$

$V_t \approx -D_t$

Note that we made abstraction of the relation of the value-function with a policy. Previous derivation only holds in case of navigating straight in the direction of the camera. From a general policy iteration point of view, this would mean that a state with closer objects and lower depth corresponds to a higher value function so it is preferred over a state with objects further away. This is of course not what we want.

Alternatively, we could assume the absolute depth as a **value-signal**. This means according to the Bellman equation that a decrease in depth, for instance by navigating towards an object, corresponds to a positive reward. For now, we leave out the discount factor γ, in order to avoid clutter in the equation.

$D_t^{'} = r_t + D_{t+1}^{'} \Rightarrow r_t = D_t^{'} - D_{t+1}^{'}$

This should not come as a surprise as the depth is then seen as the amount of +1 rewards per traveled distance in that direction up until the moment of collision. If the agent learns to focus on the short term reward, the agent will be drawn towards collisions rather than the other way around.

Obstacle avoidance is a reactive behavior. This means that the optimal control should be predictable given solely current view of the robot. This holds in cases with little drift and large enough field-of-view. Using a myopic agent simplifies the use of depth as a reward signal in reinforcement learning

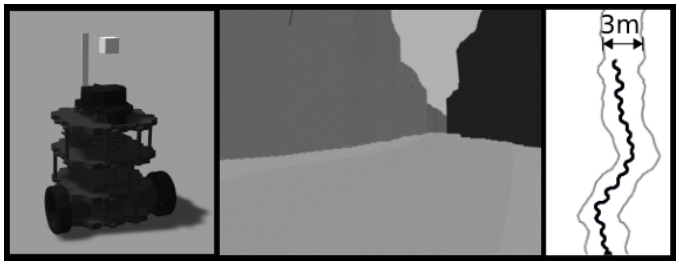


Fig. 3. Canyon example. Left: Turtlebot, middle: RGB input, right: top-down view of successful trajectory.

significantly. In this case the agent only cares about the immediate reward, picking actions that make the maximum reward most likely. In this setting the value-function corresponds to the immediate reward, in our case the depth map at the future time step. This does also resolve the contradiction explained above.

$Q(I_t, a_t) = r_t = D_{t+1}$

In this case the model learns to predict the depth seen at the next frame D_{t+1} given current frame I_t and proposed action a_t. In this paper we provide the future depth with a LiDAR at the next time step. However, the depth map could also be provided by a CNN depth-estimator as the label is allowed to be noisy. In that case the future depth network can be learned from the depth prediction on the next frame and the learning becomes fully monocular.

Depth-Q-Net

In figure 1, you can see the architecture of the network. A mobilenet-0.25 [14] convolutional neural network extracts Imagenet-pretrained [15] features from the current view of the robot. The action is concatenated to the extracted feature before it is fed to a fully-connected prediction layer of 4096 nodes and 1 fully-connected output layer of 26 nodes, corresponding to 26 depth bins from the LiDAR smoothed over 4deg from a forward field-of-view of 104deg.

The prediction layers are trained on batches of experiences in a supervised fashion with an absolute loss.

The policy is extracted by evaluating the future depth predictor on the current frame concatenated with different actions in one forward pass (see figure 2). The policy selects the action for which the predicted future depth scan has the maximum minimum depth:

$\pi(I_t) = argmax_{a_t}(min(D_{t+1}(I_t, a_t)))$

This means that the policy only focuses on the closest obstacle and takes the action that makes this closest obstacle as far as possible. This allows us to simplify the training of the future depth prediction in the sense that we only care about the closest depth. Therefore we clip the depth in our experiments at 2m.

Baseline: Coll-Q-Net

In order to explore the feasibility of using depth rather than collisions as self-supervised reward signal, we implement a similar baseline model named Coll-Q-Net. The DNN models a value-function that approximates the probability of a collision within the next H frames given current input frame and an action, similar to [12].

$Q(I_t, a_t) = \sum_{k=1}^{H} P(collision_{t+k}|I_t, a_t)$

In our experiments the horizon H is taken as 5 time steps.

The prediction layers of figure 1 contain one fully-connected layer of 4096 nodes and one fully-connected layer of 25 nodes. This results in approximately the same amount of parameters as the Depth-q-net. The fully-connected output layer has one node which contains a sigmoid activation function in order to map the output within the range of $[0, 1]$ to represent a probability. The prediction layers are trained with a cross-entropy loss. Experiments indicated the cross-entropy loss to be more stable than the mean-squared error or absolute loss.

The policy is extracted in a similar fashion, evaluating the action-value-function for a batch of actions as visible in figure 2. The action with the lowest probability of collision in the near future is selected:

$\pi(I_t) = argmin_{a_t} Q(I_t, a_t)$

Training

The prediction layers of the Depth-Q-Net and Coll-Q-Net are trained in an off-policy manner. This is beneficial as a single dataset can be collected from which multiple models can be trained in parallel, avoiding the need for data collection during training.

All experiences (I_t, a_t, y_t, d_{t+1}) are saved in a dataset and used to train the models in a supervised fashion. At each collision, the simulated environment is restarted and the last H frames get a future collision label $y_t = 1$, similar to the previous work of Kahn et al. [12]. The Depth-Q-Net does not require any collisions, therefore the last H frames of each run are discarded when training Depth-Q-Net. The data is collected by an exploration policy that randomly picks a continuous action (yaw-turn) in the range [-1:1] according to an Ornstein-Uhlenbeck process [16] to impose temporal correlation.

Although it did not have a big influence, it appeared that keeping the feature extracting part of the Mobile-0.25 net fixed was beneficial for the robustness against overfitting.

IV. EXPERIMENTS

We first discuss the simulated environment after which the results follow.

Environment

The experiments are done on a small turtlebot burger in a simulated canyon made with ROS [17] and Gazebo [18]. The simulated canyons are generated randomly during data collection. You can see an example of such a canyon in figure 3. The camera is set up on the same axis as the laser scan in order to ensure that the field-of-views between the LiDAR and the camera are aligned. The camera works at 10fps and provides frames of size 410x308 covering a field-of-view of 104deg.

The goal of the policy is to navigate the turtlebot through unseen canyons while it is driving at a fixed speed (0.5m/s)

TABLE I

ONLINE PERFORMANCE IN SIMULATED CANYON

Data	Average Distance(std)		Success rate(std)		Imitation loss(std)		Cross-Entropy(std)	Abs Diff(std)
#runs	CQN	DQN	CQN	DQN	CQN	DQN	CQN	DQN
50	1.07 (1.72E-05)	**1.68 (0.113)**	0.0 (0.0)	0.0 (0.0)	**1.03 (9.54E-03)**	2.42 (9.32E-02)	0.57 (0.128)	0.08 (3.91E-03)
100	1.04 (1.28E-04)	**4.91 (0.458)**	0.0 (0.0)	**0.7 (0.5)**	**1.22 (1.13E-02)**	1.50 (1.23E-02)	0.75 (3.09E-02)	0.08 (2.33E-03)
200	2.69 (1.28)	**12.19 (1.03)**	0.0 (0.0)	**8.0 (1.4)**	1.80 (0.307)	**0.97 (3.82E-02)**	0.85 (0.269)	0.06 (2.91E-03)
500	6.72 (0.937)	**21.96 (0.247)**	2.7 (1.2)	**20.0 (0.0)**	1.46 (6.59E-02)	**0.60 (3.54E-02)**	0.36 (9.37E-02)	0.05 (8.67E-04)
700	8.32 (1.31)	**22.21 (0.106)**	4.7 (2.1)	**20.0 (0.0)**	1.25 (6.15E-02)	**0.56 (2.61E-02)**	0.26 (2.19E-02)	0.05 (2.43E-03)
900	14.14 (3.04)	**22.30 (0.153)**	11.3 (3.8)	**20.0 (0.0)**	1.04 (0.107)	**0.55 (3.66E-02)**	0.18 (4.27E-02)	0.04 (1.19E-03)

by steering with the yaw velocity $[-1, 1]$ for a collision free distance of 15m. At test time, we only use 3 quantization levels for possible actions: -1, 0, 1. The networks are tested in 20 canyons unseen in the training data. The only difference in training Depth-Q-Net and a Coll-Q-Net is the learning-rate (0.1 and 0.01) and the selection of the loss (absolute and cross-entropy).

More detailed information on the training datasets and hyperparameters can be found in table II and the appendix.

Results

We want to investigate the benefit of using depth as a self-supervised reward signal over collisions. Besides the benefit of avoiding the need for real collisions, we are curious if the reward signal leads to more robust training for instance in the setting of having less training data.

Figure 5 shows the convergence of the absolute loss for the Depth-Q-Net and the cross-entropy loss for the Coll-Q-Net on both training and validation data for networks trained on different sizes of data.

Experience showed how not only the validation loss but especially the variance of the validation loss over a batch demonstrates potential overfitting. For the Depth-Q-Net this is visible for a dataset of 100 runs or less while for the Coll-Q-Net this is already visible at a dataset of 200 runs.

Table I gives details on the performance of the different networks. The standard deviation is calculated on 3 networks initialized with different seeding. The performance of each network is evaluated in 20 canyons. This results in an average collision free distance as well as a success rate (number of times the distance was more than 15m). Because this work solely focuses on the task of collision avoidance we do not evaluate on the time spend to travel a certain distance. The canyon however is made small enough to avoid spinning on a local spot.

The imitation loss is calculated as the MSE between the action picked by the network and the action provided by a heuristic based on the behavior arbitration algorithm taking depth as input[11]. The value is added as it gives a better measure on how the performance appears similar to an expert.

Although the reward signal is of a higher dimension and the networks are trained on less data (leaving out the collision), our method, Depth-Q-Net outperforms the baseline, Coll-Q-Net significantly on all different evaluation strategies when trained on a large enough dataset (500 runs).

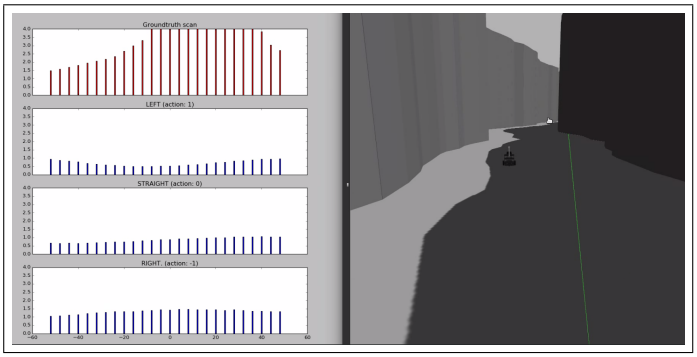


Fig. 4. Depth-Q-Net qualitative result. Left in red: ground truth scan, left in blue: future depth for left turn(up), straight(middle) and right turn(down). Right: position of robot in canyon.

In case there is not enough data, the Depth-Q-Net could still manage to pass through the canyon a number of times when trained solely on 200 or 100 runs. This is not the case for the baseline Coll-Q-Net. Although the imitation loss is less bad for the Coll-Q-Net when trained on 50 or 100 runs, the network could not drive much more than 1m without a collision.

Figure 4 gives a snapshot of the evaluation of a Depth-Q-Net in the canyon. On the left you can see in red the ground-truth lazer scan. For each action, the future depth is predicted by the network. As the Turtlebot is slightly heading to the left wall, there is an increase in overall depth for turning to the right and a decrease for turning to the left. Depth-Q-Net is only trained to predict values up until 2m while the actual depth is at 4m. Although this is far from accurate in absolute values, it is good enough to extract a working policy from.

V. CONCLUSION

In this work we explore the feasibility of using depth as a self-supervised value/reward-signal instead of collisions detected by an IMU for learning collision avoidance (as in [12]). We demonstrate that the use of depth not only leads to better performance (less collisions), it also requires less data. This is because the labels of collisions are more sparse, giving relevant feedback only at the end of each run while the Depth-Q-Net can be trained on any consecutive pair of images.

In future work the depth could be provided with a monocular depth-estimation network, in which case it would not require any additional sensors besides the RGB camera. The

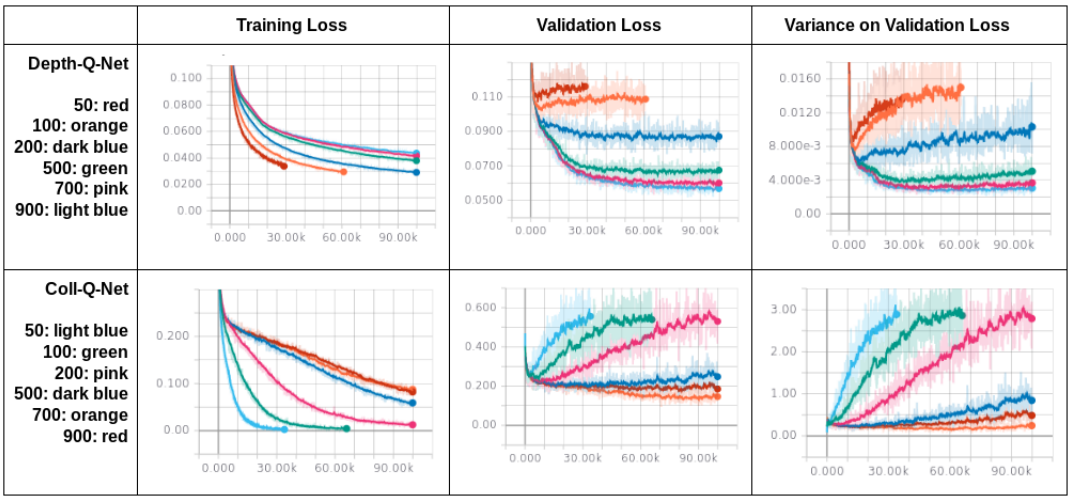


Fig. 5. Convergence difference for different networks trained on different sizes of datasets.

TABLE II
DETAILS DIFFERENT SIZES OF DATASETS

# runs	# samples	# without collision
50	2214	1858
100	4258	3945
200	8854	7797
500	23361	20517
700	33303	29152
900	42424	38032

Depth-Q-Net will learn to predict the estimated depth at the next step given current RGB image and proposed action.

On the other hand, a real-world experiment could even further demonstrate the benefit of training a Depth-Q-Net over a Coll-Q-Net as acquiring data in the real world is much more difficult especially if it requires collisions. Unfortunately lack of time did not allow us to include these experiments although the simulated results already demonstrate clearly the feasibility of our method.

Although depth-value-signals are probably not the full solution for monocular collision avoidance, it can be interesting to add in the form of intrinsic motivation for general autonomous robots provided with a camera.

In the current work we only look at the next frame. With a recurrent neural network more aggressive behaviors might be learned where planning over multiple time steps is required.

ACKNOWLEDGEMENTS

This work was supported by the CAMETRON research project of the KU Leuven (GOA) and the Omnidrone research project (FWO-SBO).

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, 10 2015.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2017.

[3] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive Mapping and Planning for Visual Navigation," *CVPR*, 2017.

[4] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, and D. London, "Learning To Navigate In Complex Environments," *ICLR*, 2017.

[5] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *NIPS*, 6 2014.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *ICML*, 2 2016.

[7] S. Daftry, J. A. Bagnell, and M. Hebert, "Learning Transferable Policies for Monocular Reactive MAV Control," *International Symposium on Experimental Robotics (ISER)*, 8 2016.

[8] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, pp. 661–667, 7 2016.

[9] S. Ross, N. Melik-Barkhudarov, S. K. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and H. Martial, "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," *ICRA*, pp. 1765–1772, 2013.

[10] J. Michels, A. Saxena, and A. Y. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," *ICML*, no. 22, 2005.

[11] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. V. Eycken, "CNN-based Single Image Obstacle Avoidance on a Quadrotor," *ICRA*, 2017.

[12] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation," *International Conference on Robotics and Applications(ICRA)*, 2018.

[13] M. A. Anwar and A. Raychowdhury, "NAVREN-RL: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images," 7 2018.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *Arxiv 1704.04861*, 2017.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 12 2015.

[16] G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," *Physical Review*, vol. 36, pp. 823–841, 9 1930.

[17] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA*, 2009.

[18] Open Source Robotics Foundation, "Gazebo," 2018.

VI. Appendix

Training deep neural networks can be tedious. In order to reproduce the results we share our hyperparameters. They can also be found in the code itself on the project page: kkelchte.github.io/depth-q-net.

- Dropout of 0.5 after the output of the Mobilenet-0.25.
- Batch size of 64.
- Weight decay of 4e-5.
- Adadelta optimizer.
- Xavier initialization.
- Ended after 1000 epochs ($\approx$ 3h) .

Coll-Q-Net had to be trained with a lower learning rate (0.01) than Depth-Q-Net (0.1). As mentioned in the paper, Coll-Q-Net trained best with a cross-entropy loss and Depth-Q-Net with an absolute difference.

Automatic generation of ground truth for the evaluation of obstacle detection and tracking techniques

Hatem Hajri*, Emmanuel Doucet*†, Marc Revilloud*, Lynda Halit*, Benoit Lusetti*, Mohamed-Cherif Rahal*

Automated Driving Research Team, Institut VEDECOM, Versailles, France
†*InnoCoRe Team*, Valeo, Bobigny, France

***Abstract*— As automated vehicles are getting closer to becoming a reality, it will become mandatory to be able to characterise the performance of their obstacle detection systems. This validation process requires large amounts of ground-truth data precisely describing the pose and kinematics of the obstacles surrounding the vehicle. The creation of such datasets currently requires the lengthy and arduous process of manually interpreting sensor data. In this paper, we propose a novel methodology to generate ground-truth kinematics datasets for specific objects in real-world scenes. Our procedure requires no annotation whatsoever, human intervention being limited to sensors calibration. We present the recording platform which was exploited to acquire the reference data, then fully describe our data generation process. A detailed and thorough analytic study of the propagation of errors in our procedure is also performed. This allows us to provide detailed precision metrics for each and every data item in our datasets.**

The main contributions of this paper reside in the data-generation methodology, the analysis of the error propagation, and in the creation of a new dataset.

I. Introduction

Object detection and tracking both play a crucial role in autonomous driving. They are low-level functions upon which many other increasingly high-level functions are built. These functions include Intention prediction, Obstacle avoidance, Navigation and planning. Being depended on by so many functions, the task of obstacle detection and tracking must be performed with a high level of accuracy and be robust to varying environmental conditions. However, the generation of ground truth data to evaluate obstacle detection and tracking methods usually involves manual annotation, either of images, or of LIDAR point clouds [1][2][3].

This paper showcases a method which takes advantage of the multiplication of autonomous driving platform prototypes in research structures to generate precise and accurate obstacle ground truth data, without requiring the usual phase of painstaking manual labelling of raw data.

Firstly, the methodology applied to generate this data will be described in general terms, and some specific technical topics such as the sensors time-synchronisation method used to collect data will be presented. Then, a thorough analysis of errors propagation is performed. Finally some plots of the collected data are given together with potential applications.

II. General method description

The proposed method requires two or more vehicles to generate obstacles dynamics ground truth data. The first vehicle -the ego-vehicle- is equipped with a high-precision positioning system (for example GNSS with Real Time Kinematics (RTK) corrections coupled with an IMU), and various environment perception sensors (for example LIDARs, cameras or RADARs). The other vehicles -the target vehicles- only need to be equipped with a high-precision positioning system, similar to that of the ego-vehicle. By simultaneously recording the position and dynamics of all vehicles, it is possible to express the kinematics of all equipped vehicles present in the scene, in the ego-vehicle frame of reference, and therefore to generate reference data for these vehicles. This reference data can then be used to evaluate the performance of obstacle detection methods applied to the environmental sensors data collected on the ego-vehicle.

III. Data collection setup

This section, presents in details the set of sensors which were available for the data collection, and details the method applied to ensure the synchronicity of the recording process taking place in different vehicles. Three vehicles were used during this data collection campaign : the ego-vehicle was a Renault Scenic equipped to acquire environment perception data with a high accuracy. The target vehicles were Renault ZOEs, modified to be used as autonomous driving research platforms, and therefore equipped with precise positioning systems.

A. Ego-vehicle perception sensors

To record perception data, the ego-vehicle is equipped with two PointGrey 23S6C-C colour cameras, a Velodyne VPL-16 3D laser scanner (16 beams, 10Hz, 100m range, 0.2° horizontal resolution), a cocoon of five Ibeo LUX laser scanners (4 beams, 25Hz, 200m range, 0.25° horizontal resolution) covering a field of view of 360° around the vehicle.

The cameras are positioned inside the car, behind the windscreen with a baseline of approximately 50cm. The Velodyne VLP-16 is positioned on the roof of the vehicle at a position and height that minimise the occlusion of the laser beams by the vehicle body. The Ibeo LUX are all mounted at the same height of approximately 50cm, two on each front wing (one pointing forward, one to the side), and one at the back, pointing backwards (see Figure 2).

Fig. 1. The perception vehicle used : two ibeo LUX, the VLP16, GNSS antenna and the precision odometer are visible

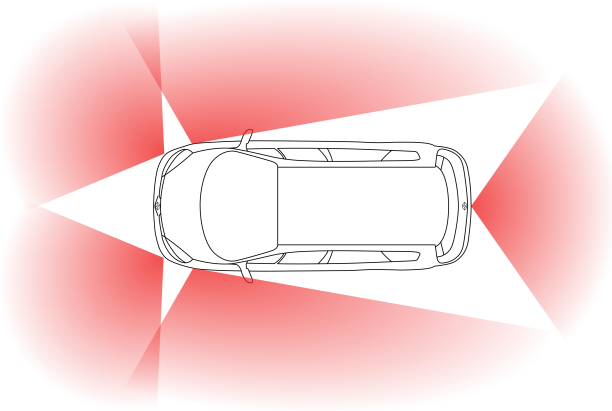


Fig. 2. Ibeo LUX cocoon setup

B. Precise localisation system

The accuracy of the data generated using our method relies entirely on the accuracy of the vehicles' positioning systems. Therefore, each vehicle was equipped with state of the art positioning sensors : a choke-ring GNSS antenna feeding a GNSS-RTK receiver coupled with a high-grade, fibre optic gyroscopes-based iXblue Inertial Measurement Unit. Additionally, the perception vehicle is equipped with a high-accuracy odometer mounted on the rear-left wheel, while the target vehicles are equipped with a Correvit® high-accuracy, contact-less optical odometer. The data emanating from these sensors is fused using a Kalman Filter-based robust observer which jointly estimates the IMU and external sensors biases. The performance of this system can be further improved in post-processing by employing accurate ephemeris data and smoothing techniques. Table I provides an overview of the combined performance of our positioning system and of the the aforementioned post-processing.

C. Time synchronisation

One of the biggest challenges of performing data collection distributed across multiple vehicles is to precisely synchronise the clocks used for time-stamping the data in each platform. This is especially true when acquiring data in high-speed scenarios. Highway scenarios, in which the absolute value of the relative velocity of vehicles may reach 70m/s, require the synchronisation of the vehicle clocks to be at least accurate to the millisecond. This inaccuracy induces an incompressible positioning error, which adds to that of our positioning system (see Subsection VI-A).

To achieve such a precise synchronisation, a Network Time Protocol (NTP) server fed with the pulse-per-second (PPS) signal provided by our GNSS receivers was installed in each vehicle to synchronise the on-board computers in charge of recording all the data. Prior to any recording session, the NTP servers and computer clocks were allowed 12 hours to converge to a common time.

TABLE I

POSITIONING SYSTEM PERFORMANCE

	Heading (deg)	Roll/Pitch (deg)	Position X,Y (m)	Position Z (m)
Nominal GNSS signal	0.01	0.005	0.02	0.05
60 sec GNSS outage	0.01	0.005	0.10	0.07
300 sec GNSS outage	0.01	0.005	0.60	0.40

IV. SENSORS CALIBRATION

Generating ground truth data requires very accurate sensors calibration. Given the difficulty of calibrating LIDAR sensors relatively to cameras[1], we propose the following calibration pipeline : first, the positioning system is calibrated, then the cameras are calibrated intrinsically, and finally the rigid transformations relating cameras and LIDARs to the vehicle frame are estimated. To take into account any potential effect of the sensors positioning on their calibration, all sensors intrinsic parameters estimation processes take place once the sensors are installed in the vehicle.

A. Positioning system calibration

The calibration of the positioning system consists in calculating the position and attitude of all positioning sensors (GNSS antenna, optical odometer) in the frame of reference of the IMU. After a phase of initialisation during which the vehicle remains static to allow the estimation of all sensors biases and the convergence of the RTK, the vehicle is manoeuvred. By comparing the motion information emanating from each sensor and comparing it to that of the IMU, one is then able to determine the rigid transformation between the said sensor and the IMU.

B. Cameras calibration

To estimate the intrinsic parameters of our cameras, and the relative pose of our stereo pair, we sweep the space in front of the cameras at three different depths, making sure to cover the whole field of view of each camera. We then use a mixture of Geiger's checkerboard pattern detector [4] and of the sub-pixellic corner detector from openCV to extract the corners of the checkerboard. These are then fed to openCV's `stereoCalibrate` function to jointly estimate

the intrinsic parameters of each camera and their relative pose. The set of parameters thus obtained typically yields re-projection errors of less than 0.3 pixel.

The position and orientation of our cameras are obtained in a semi-automatic fashion. It involves minimising the reprojection error of a known pattern in the images. Using laser tracers, the position of the cameras in the vehicle, and the position and orientation of the vehicle relative to the ground pattern are precisely measured. The orientation of the cameras in the vehicle frame is then estimated by minimising the reprojection error of the pattern characteristic points in the image using a Levenberg-Marquardt minimisation method.

C. LIDARs calibration

1) Velodyne VLP-16 calibration: The objective of the Velodyne to IMU calibration process is to determine the rigid transformation $T_{Vel \to IMU}$ between the Velodyne reference frame and that of the IMU. Our Velodyne to IMU calibration process is fully automated. Using Iterative Closest Point, we match the 3D point clouds acquired during the calibration manoeuvre one to another to generate a trajectory. Then, the pose of the IMU is re-sampled to the timestamps of the Velodyne scans. The estimation of $T_{Vel \to IMU}$ knowing both trajectories is similar to the *hand-eye calibration problem*. This estimation is performed using a non-linear optimisation process applied to 1000 pose samples [5]. This rigid transformation estimate is then refined by repeating the process, using $T_{Vel \to IMU}$ and the linear and angular velocities of the vehicle to correct the motion-induced distorsion of the Velodyne point clouds. This process usually converges in just one iteration.

2) Ibeo LUX: The calibration of an Ibeo LUX cocoon is slightly more complicated than that of a single Velodyne, as it involves simultaneously calibrating all the sensors. Indeed, calibrating each LUX separately will almost certainly result in a poorly consistent point cloud when aggregating clouds from all sensors. Likewise, precisely calibrating one sensor, and then calibrating all sensors relative to their neighbour will also lead to such poor results. A simple way to ensure the global coherence of the cocoon calibration is to use the point cloud from a calibrated Velodyne as a reference, and to calibrate all Ibeo LUX sensors relative to this spatially coherent reference.

V. Ground-truth data generation

The objective of the ground-truth generation process is, at every instant, to precisely and accurately describe the position, orientation and kinematics of an observed obstacle in the frame of reference of the ego-vehicle. This data can then be used to estimate the accuracy and precision of an obstacle detection method or sensor.

A. Notations

Let $X_i^k = (x, y, v_x, v_y, \psi)_i^k$ be the state of vehicle i, with $(x, y)_i^k$ the position of its reference point, $(v_x, v_y)_i^k$ its velocity vector, and ψ_i^k its yaw angle, all expressed in the frame of reference k.

In the rest of the paper, $\cdot_e$ and $\cdot_t$ respectively denote state variables of the ego and of the target vehicle, and $\cdot^{UTM}$ and $\cdot^{ego}$ respectively denote a variable expressed in the Universal Transverse Mercator and in the ego-vehicle frame of reference.

B. Processing

Generating a set of obstacle ground truth data from high accuracy positioning recordings is a two step process :

- generate the relative position and dynamics of the obstacles relative to the ego-vehicle,
- generate data carrying obstacle semantics from the previously generated data.

For each sensor recording, a whole set of ground truth data is generated, so as to provide ground truth synchronised with the sensor data. At each sensor data timestamp, the position and dynamics of each vehicle are estimated from the positioning system recording using temporal splines interpolation.

From this, simple kinematics and velocity composition formulae allow the computation of the relative position and dynamics of the target vehicles in the ego-vehicle frame of reference :

$$\begin{bmatrix} x \\ y \end{bmatrix}_t^{ego} = R(-\psi_e^{UTM}) \begin{bmatrix} x_t - x_e \\ y_t - y_e \end{bmatrix}^{UTM} \tag{1}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix}_t^{ego} = R(-\psi_e^{UTM}) \begin{bmatrix} v_{xt} - v_{xe} + \dot{\psi}_e(y_t - y_e) \\ v_{y_t} - v_{y_e} - \dot{\psi}_e(x_t - x_e) \end{bmatrix}^{UTM} \tag{2}$$

$$\psi_t^{ego} = \psi_t^{UTM} - \psi_e^{UTM} \tag{3}$$

Where $R(\alpha)$ denotes a rotation in SO_2 of angle α

C. Exploitation

The reference relative positioning data thus obtained can then be used to generate ground truth perception data. One can for example generate the bounding box of the target vehicle in the ego-vehicle frame of reference to evaluate the performance of LIDAR or image-based object detection and tracking algorithms. Another possibility is to use 3D models of the target vehicles and to project them in the camera images to automatically generate a partial image segmentation.

VI. Uncertainty propagation analysis

In this section, a sensitivity analysis of the proposed ground truth generation process is performed, to characterise the accuracy and precision of the generated data, depending on the performance of our positioning systems, and the clock shift between the vehicles.

The inputs of the generation process are made of position, velocity and heading estimates provided by a GNSS-INS fusion

system. These can be modelled as independent, Gaussian random variables[6].
Therefore, the position, velocity and yaw angle are treated separately, so as to limit the calculation hurdle.

A. Position

Equation (1) yields :

$$\left[\begin{array}{c} x \\ y \end{array} \right]_t^{ego} = F(dx^{UTM}, dy^{UTM}, \psi_e^{UTM})$$

with :

$$F(dx, dy, \psi_e) = \left[\begin{array}{c} dx\cos\psi_e + dy\sin\psi_e \\ dy\cos\psi_e - dx\sin\psi_e \end{array} \right].$$

Lemma 1: Let Ω be a Gaussian random variable such that $\Omega \sim \mathcal{N}(m_\Omega, \sigma_\Omega^2)$. Then:

$$\mathbb{E}(\cos(\Omega)) = \cos(m_\Omega)\ e^{-\sigma_\Omega^2/2}$$
$$\mathbb{E}(\sin(\Omega)) = \sin(m_\Omega)\ e^{-\sigma_\Omega^2/2}$$

Proof: Using the explicit expression of the characteristic function of a Gaussian variable yields:

$$\mathbb{E}(\cos(\Omega) + i\sin(\Omega)) = \mathbb{E}(e^{i\Omega}) = e^{i\,m_\Omega - \sigma_\Omega^2/2}$$

The real and imaginary part of this expression yield the desired result. ∎

Under the assumption that $\mathrm{Var}(dx) = \mathrm{Var}(dy) = \sigma_{dx}^2$, and noting $\mathrm{Var}(\psi_e) = \sigma_\psi^2$:

$$\mathrm{Cov}(F(dx, dy, \psi_e)) = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

With :

$$\begin{aligned} a = {} & \sigma_{dx}^2 + \mathbb{E}(dx)^2\mathrm{Var}(\cos\psi_e) + \mathbb{E}(dy)^2\mathrm{Var}(\sin\psi_e) \\ & - \mathbb{E}(dx)\mathbb{E}(dy)\sin(2\mathbb{E}(\psi_e))e^{-\sigma_\psi^2}(1 - e^{-\sigma_\psi^2}) \end{aligned}$$

$$\begin{aligned} b = {} & \sigma_{dx}^2 + \mathbb{E}(dx)^2\mathrm{Var}(\sin\psi_e) + \mathbb{E}(dy)^2\mathrm{Var}(\cos\psi_e) \\ & + \mathbb{E}(dx)\mathbb{E}(dy)\sin(2\mathbb{E}(\psi_e))e^{-\sigma_\psi^2}(1 - e^{-\sigma_\psi^2}) \end{aligned}$$

$$\begin{aligned} c = {} & \frac{1}{2}\sin(2\mathbb{E}(\psi_e))e^{-\sigma_\psi^2}(1 - e^{-\sigma_\psi^2})(\mathbb{E}(dx)^2 - \mathbb{E}(dy)^2) \\ & - \mathbb{E}(dx)\mathbb{E}(dy)\cos(2\mathbb{E}(\psi_e))e^{-\sigma_\psi^2}(1 - e^{-\sigma_\psi^2}) \end{aligned}$$

Therefore, under the assumption that the variance of the position error is similar from one vehicle to another, and along North and East axes ($\sigma_x^2 = \sigma_y^2 = \sigma_{pos}^2 = \frac{1}{2}\sigma_{dx}^2$), and that the maximal distance between the ego-vehicle and an obstacle is d_{max}, we propose the following upper bound for the position error covariance matrix:

$$\begin{aligned} a &\leq 2\sigma_{pos}^2 + 2d_{max}^2(1 - e^{-\sigma_\psi^2}), \\ b &\leq 2\sigma_{pos}^2 + 2d_{max}^2(1 - e^{-\sigma_\psi^2}), \\ c &\leq \frac{3}{2}d_{max}^2(1 - e^{-\sigma_\psi^2/2}). \end{aligned}$$

B. Velocity

Equation (2) yields :

$$\left[\begin{array}{c} v_x \\ v_y \end{array} \right]_t^{ego} = G\left((dx, dy, dv_x, dv_y, \psi_e, \dot{\psi}_e)^{UTM}\right)$$

with $G(dx, dy, dv_x, dv_y, \psi_e, \dot{\psi}_e)$ equal to :

$$\begin{bmatrix} \cos\psi_e(dv_x + \dot{\psi}_e dy) + \sin\psi_e(dv_y - \dot{\psi}_e dx) \\ \cos\psi_e(dv_y - \dot{\psi}_e dx) - \sin\psi_e(dv_x + \dot{\psi}_e dy) \end{bmatrix}$$

Using the independence of the input variables, it is possible to express the covariance matrix of $G(dx, dy, dv_x, dv_y, \psi_e, \dot{\psi}_e)$ as a function of the first and second order moments of the input variables. Due to space limitations, we only give bounds for the elements of this matrix. Similarly to Section (VI-A), these bounds tend to 0 as the variances of the input variables tend to 0.

Lemma 2: Let X and Y be two independent random variables with means m_X and m_Y and variances σ_X^2 and σ_Y^2. Let Ω be a Gaussian random variable $\mathcal{N}(m_\Omega, \sigma_\Omega^2)$ independent of X, Y. Let $Z = \cos(\Omega)X + \sin(\Omega)Y$. Then :

$$\mathrm{Var}\ (Z) \leq \sigma_X^2 + \sigma_Y^2 + (1 - e^{-\sigma_\Omega^2})(|m_X| + |m_Y|)^2$$

Proof: Using Lemma 1, we have the following bound:

$$\begin{aligned} \mathrm{Var}(Z) \leq {} & \sigma_X^2 + \sigma_Y^2 + m_X^2\mathbb{E}(\cos^2(\Omega)) + m_Y^2\mathbb{E}(\sin^2(\Omega)) \\ & - e^{-\sigma_\Omega^2}(m_X^2\cos^2(m_\Omega) + m_Y^2\sin^2(m_\Omega)) \\ & - m_X m_Y \sin(2m_\Omega)e^{-\sigma_\Omega^2}(1 - e^{-\sigma_\Omega^2}) \end{aligned}$$

Using the identity $\cos^2(\Omega) = \frac{1+2\cos(\Omega)}{2}$ and again Lemma 1, we get :

$$m_X^2\mathbb{E}(\cos^2(\Omega)) - e^{-\sigma_\Omega^2}m_X^2\cos^2(m_\Omega) \leq m_X^2(1 - e^{-\sigma_\Omega^2})$$

The same identity holds with X replaced with Y and cos replaced with sin. This gives the desired bound. ∎

Let us denote the mean and variance of a Gaussian variable Z respectively by m_Z and σ_Z^2. Under the assumption that $\mathrm{Var}(dv_x) = \mathrm{Var}(dv_y) = 2\sigma_{vel}^2$ and $\mathrm{Var}(dx) = \mathrm{Var}(dy) = \sigma_{dx}^2$, we have :

$$\mathrm{Var}(dv_x + \dot{\psi}dy) = 2\sigma_{vel}^2 + \sigma_{dx}^2\sigma_{\dot{\psi}}^2 + m_{\dot{\psi}}^2\sigma_{dx}^2 + m_{dy}^2\sigma_{\dot{\psi}}^2$$

$$\mathrm{Var}(dv_y - \dot{\psi}dx) = \mathrm{Var}(dv_x + \dot{\psi}dy)$$

Let $\begin{bmatrix} a & c \\ c & b \end{bmatrix}$ be the covariance matrix of $\left[\begin{array}{c} v_x \\ v_y \end{array} \right]_t^{ego}$. Using the previous results, the upper bounds for a, b and c are:

$$\begin{aligned} a, b \leq {} & 4(\sigma_{vel}^2 + \sigma_{pos}^2\sigma_{\dot{\psi}}^2 + \dot{\psi}_{max}^2\sigma_{pos}^2) + 2d_{max}^2\sigma_{\dot{\psi}}^2, \\ & + 4(1 - e^{-\sigma_\psi^2})(v_{max} + d_{max}\dot{\psi}_{max})^2 \end{aligned}$$

$$c \leq \sqrt{a^2}\sqrt{b^2},$$

where v_{max} is an upper bound for dv_x and dv_y and $\dot{\psi}_{max}$ is an upper bound for $\dot{\psi}$.

C. Yaw angle

The variance of the relative yaw estimate ψ_t^{ego} is trivially derived from equation (3). Considering that the heading error covariance is similar in the target and ego-vehicle ($\sigma^2_{\psi_t^{UTM}} = \sigma^2_{\psi_e^{UTM}} = \sigma^2_{\psi}$), we get:

$$\mathrm{Var}(\psi_t^{ego}) = 2\sigma^2_{\psi}$$

D. Results

Using the analytic upper bounds for the covariance matrices of the generated ground truth position, speed and yaw angle data calculated in the previous sections, we can estimate the precision of the data generated using the process described in section V.
The following values will be used :

σ_{pos}	0.02	m
σ_{vel}	0.02	m.s^{-1}
σ_{ψ}	$1.75.10^{-3}$	rad
d_{max}	50	m
v_{max}	36	m.s^{-1}
$\dot{\psi}_{max}$	1	rad.s^{-1}

They are representative of the performance of our positioning system, of the distance at which obstacles become truly observable, and of the maximal dynamics of the vehicle in typical use cases.
These values, when injected in the upper bounds of the position and velocity covariance matrices yield the following results :

$$\|\mathrm{Cov}((x,y)_t^{ego})\|_F^{1/2} \leq 0.12\ m$$
$$\|\mathrm{Cov}((v_x,v_y)_t^{ego})\|_F^{1/2} \leq 0.30\ m.s^{-1}$$

These values represent the root mean square error on the position and velocity information yielded by our ground truth generation process. We stress the fact that they are obtained by taking extreme values for all variables describing the dynamics of both vehicles, which can never actually be encountered simultaneously in real life situations.

VII. Data visualisation

In this section some figures of the acquired data are given. These data were collected during a tracking scenario on a track located in Versailles France. The track has approximative length of 3.2 km and is represented in Figure 3 together with the starting and arriving points of the vehicles.
The following figures show temporal variations of the relative positions x, y, relative velocities vx, vy and orientation ψ of the tracked vehicle, as estimated by our LIDAR's firmware (in the ego vehicle frame), and the corresponding ground truth values deduced from RTK sensors as explained in the previous section. All these quantities are in the International System of Units.

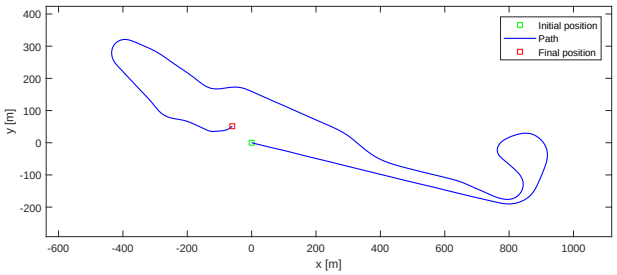


Fig. 3. Test track

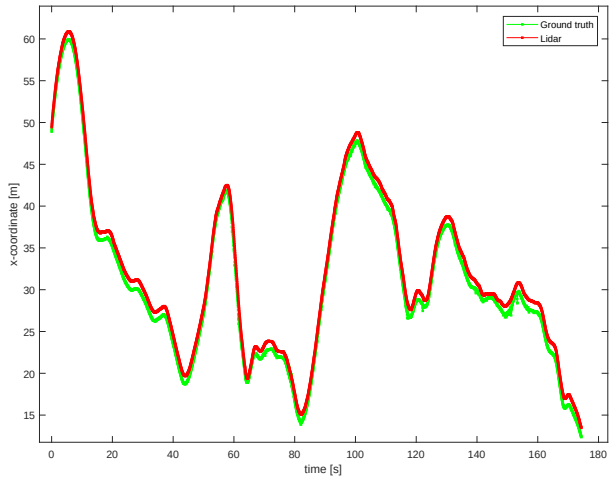


Fig. 4. Variations of x

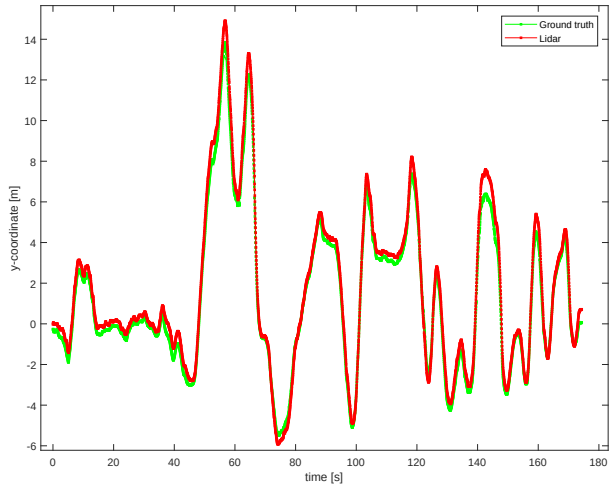


Fig. 5. Variations of y

VIII. Application to the evaluation of perception algorithms

This section gives some potential applications of ground truth data to the conception and evaluation of perception algorithms.

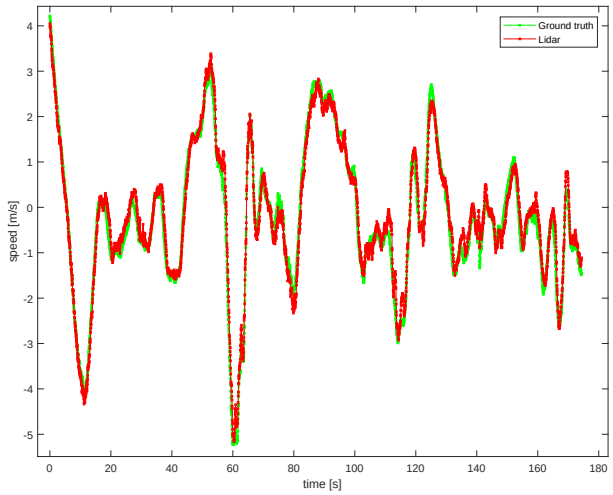


Fig. 6. Variations of vx

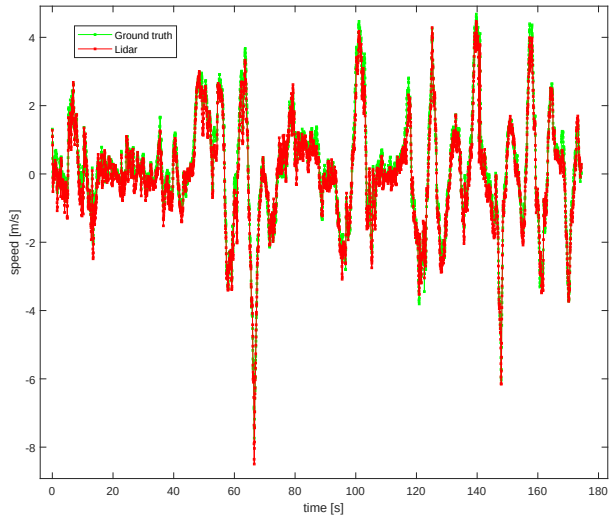


Fig. 7. Variations of vy

A. Clustering Lidar raw data:

Robust obstacle detection using sensors such as Lidar is a key point for the development of autonomous vehicles. Lidar performance mainly depends on its low level processing (the clustering methods used for its raw data, estimations of the bounding boxes, velocities of the representative points of the clusters, etc.). The methodology presented in the paper can be used to evaluate the performance of the main outputs of a Lidar raw data clustering algorithm.

B. Lidar-Radar-Camera Fusion:

Lidars, Radars and Cameras are complementary sensors. Lidars are very accurate on obstacles positions and less accurate on their velocities. On the other hand, Radars are more precise on obstacles velocities and less precise on their positions. Cameras provide images and can be used to perform classification tasks. Fusion between these sensors aims at combining the advantages of each sensor to provide permanent and more robust data (see for example [7], [8], [9], [10], [11]). In particular, the recent work [7] proposes a Lidar-Radar fusion algorithm with evaluation on the database generated in the present paper.

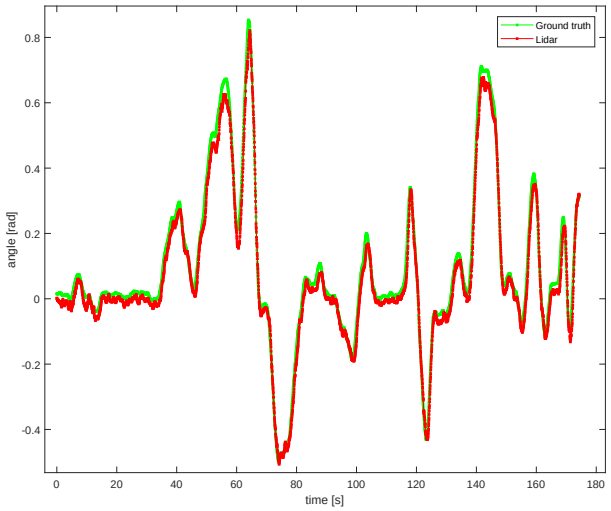


Fig. 8. Variations of ψ

C. Dynamic models and prediction of obstacles motions:

Robust tracking of obstacles detected by sensors such as Lidars, Radars and Cameras is crucial for a good functioning of autonomous cars. Kalman filters are among the most used methods to track obstacles. These methods are usually coupled to preset models such as constant velocity, acceleration or curvature. Ground truth allows to learn true dynamic models of obstacles using statistics, neural networks etc. The learnt models are to be compared with the preset ones.

IX. Summary and Future Developments

In this paper, we have presented a method for automatically generating sets of ground truth data to support advances in the field of obstacle detection and tracking. We hope this methodology will help contributors of this area of research to challenge their approaches, and contribute to the development of robust and reliable algorithms. In the future, we intend to propose a complete benchmark to unify the usage of this dataset and the performance estimation of obstacle detection and tracking techniques.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] H. Yin and C. Berger, "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–8.

[4] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic calibration of range and camera sensors using a single shot," in *International Conference on Robotics and Automation (ICRA)*, 2012.
[5] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, Aug 1998.
[6] X. Niu, Q. Chen, Q. Zhang, H. Zhang, J. Niu, K. Chen, C. Shi, and J. Liu, "Using allan variance to analyze the error characteristics of gnss positioning," *GPS Solutions*, vol. 18, no. 2, pp. 231–242, Apr 2014. [Online]. Available: https://doi.org/10.1007/s10291-013-0324-x
[7] H. Hajri, L. Halit, B. Lusetti, and M.-C. Rahal, "Real time Lidar and Radar high-level fusion for obstacle detection and tracking," *submitted paper*, 2018.
[8] C. Blanc, B. T. Le, Y. Le, and G. R. Moreira, "Track to Track Fusion Method Applied to Road Obstacle Detection," *IEEE International Conference on Information Fusion*, 2004.
[9] M. S. T. G. D. Gohring, M. Wang, "Radar/lidar sensor fusion for car-following on highways," *IEEE Int. Conf. Autom. Robot. Appl.*, pp. 407–412, 2011.
[10] R. O. C. García and O. Aycard, "Multiple sensor fusion and classification for moving object detection and tracking," *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 2, pp. 525–534, 2016. [Online]. Available: http://dx.doi.org/10.1109/TITS.2015.2479925
[11] A. Rangesh and M. M. Trivedi, "No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & Lidars," *https://arxiv.org/pdf/1802.08755.pdf*, 2018.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Autonomous navigation using visual sparse map*

Soonhac Hong, *Member, IEEE,* and Hui Cheng, *Senior Member, IEEE*

***Abstract*— This paper presents an autonomous navigation system using only visual sparse map. Although a dense map provides detail information of environment, most information of the dense map is redundant for autonomous navigation. In addition, the dense map demands the high cost for storage, transmission and management. To tackle these challenges, we propose the autonomous navigation using a visual sparse map. We leverage visual Simultaneous Localization and Mapping (SLAM) to generate the visual sparse map and localize a robot in the map. Using the robot position in the map, the robot navigates by following a reference line generated from the visual sparse map. We evaluated the proposed method using two robot platforms in indoor environment and outdoor environment. Experimental results show successful autonomous navigation in both environments.**

I. INTRODUCTION

Autonomous navigation is an essential component for a robot to reach a goal location. For autonomous navigation, dense maps have been typically used [4 - 15]. However, there are a couple of challenges of dense map based autonomous navigation. First, most points of a dense map are redundant for localization and navigation. Second, the dense map needs to be updated periodically if environment changes. Thus, high-cost map management and computation follows and a high transmission bandwidth is required to update the dense map. Third, a large memory is needed to store the dense map as the map size increases.

To tackle these challenges of dense map based autonomous navigation, we propose an autonomous navigation system using visual sparse map as shown in Fig. 1. The autonomous navigation system using visual sparse map has two phases; 1) map generation and 2) autonomous navigation.

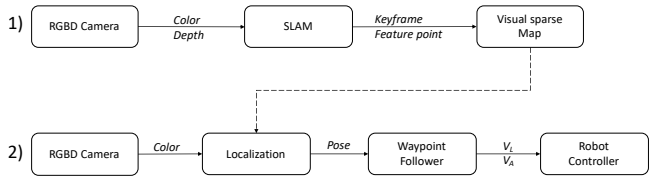


Figure 1. Overview of autonomous navigation using visual sparse map

In the map generation phase, color images and depth images from a RBG-D camera are used to generate a visual sparse map by Simultaneous Localization and Mapping (SLAM). As the visual sparse map includes only visual feature points and keyframes as shown in Fig. 2, the map size can be reduced considerably. Each visual feature point has the 3D position of the visual feature point. Each keyframe has 3D position and 3D orientation.

*This work has been supported by JD.com.
Soonhac Hong and Hui Cheng are with JD.com, Mountain View, CA 94043 USA (e-mail: soonhac.hong@ jd.com and hui.cheng@jd.com).

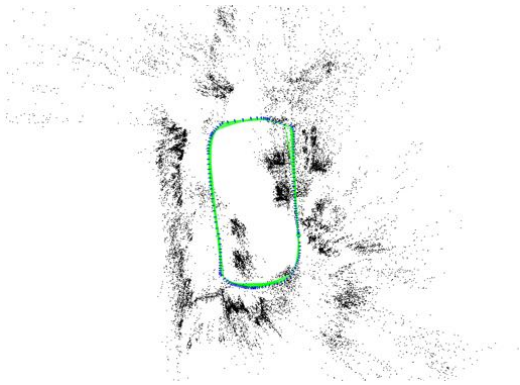


Figure 2. Example of visual sparse map (Datase I). Blue points represent the keyframe of the map. Green lines represent the visibility among keyframes. Black points represent visual feature points.

In the autonomous navigation phase, only color images are used for localization. A SLAM algorithm computes the robot pose using a color image and the visual sparse map. Using the robot pose and keyframes in the visual sparse map, the waypoint follower computes a translation velocity and a angular velocity to enable the robot to follow the reference line, a list of keyframes in the map.

This paper is organized as follows. Section II reviews related works. Section III briefly describes the SLAM algorithm. Section IV explains the waypoint follower. Section V presents experimental results and the paper is concluded in Section VI.

II. RELATED WORK

A couple of maps have been introduced for autonomous navigation. Metric map is one of the popular maps for autonomous navigation. In a metric map, positions of landmarks or objects in an environment are stored in a map with respect to a global coordinate system. Metric map can be classified by continuous map and discrete map [1]. While the former represents the environment using lines or polygons [2, 3], the latter represents the environment using cells, points, Surfel, Voxel, and features. Discrete map can be classified as dense map and sparse map according to map density. Cell, point, Surfel and Voxel have been used for dense map and features have been used for sparse map.

Occupancy grid map is a typical map using cells for autonomous navigation [4 - 6]. Each cell of an occupancy grid map represents whether a space is occupied by objects or not. A path for navigation is planned on the occupancy grid map. However, the occupancy grid map typically represents the environment in 2D space. For 3D space, a point cloud map has been used [6 - 10]. As the point cloud map densely represents

the environment as many points, the size of the point cloud substantially increases as the map area grows. To reduce the size of point cloud map, Surfel [11, 12] and Voxel [4, 13 - 15] are introduced. However, Surfel and Voxel still need high computational cost for post-processing for generating Surfel and Voxel. In addition, most information of the dense map is redundant for autonomous navigation. Thus, a sparse map has been proposed.

The sparse map can be represented as features (e.g. visual feature descriptors) [16 - 21]. As each visual features can be generated from corners or blobs in the image, the number of visual feature points is much smaller than the number of points in the point cloud map. However, most works on sparse map have focused on mapping and localization. There has been a little attention for autonomous navigation using a sparse map [29]. Although [29] uses a spare map generated by Harris corner detector, it uses a point cloud map not visual feature descriptors for a map. Thus, this paper presents the autonomous navigation system using visual sparse map.

III. Mapping And Localization

We leverage ORB-SLAM2 [22] for building a visual sparse map and localization. This section gives the brief summary of mapping and localization of ORB-SLAM2 and additional methods we implement for the proposed system. Further details of ORB-SLAM2 can be found at [22].

A. Mapping

ORB-SLAM2 consists of three modules; 1) Tracking, 2) Local mapping, and 3) Loop closing. When a new image is captured, the tracking module checks if a local map is available. If there is no map available, a local map is initialized. If the local map is available, the tracking module predicts a relative pose between the new image and the local map using the motion model. If the motion model is not available, the relative pose is predicted using visual odometry with respect to the last keyframe. If neither motion model nor visual odometry predicts the relative pose, relocalization predicts the relative pose. Relocalization finds similar keyframes using visual vocabulary in the map and estimates the relative pose to the most similar keyframe. If the relative pose is successfully estimated by motion model, visual odometry or relocalization, the relative pose is refined with the local map. If the relative pose of the new image is successfully computed, the tracking module determines if the new image is a new keyframe. If the number of matched points between the current image and the last keyframe is smaller than a threshold, the new image is determined as the new keyframe.

If a new keyframe is generated by the tracking module, the new keyframe is added to the local map. Given the new keyframe, the local map module optimizes the local map using a local Bundle Adjustment (BA). To limit the size of the local map, the local map deletes redundant keyframes in order to maintain a compact local map. If a keyframe has 90% of the map points which has been seen in at least other three keyframes, the keyframe is determined as a redundant keyframe and deleted in the local map.

Given the new keyframe, the loop closing module checks if the new keyframe is the revisited image. The loop closing module recognizes the revisited place using a place recognition database consisting of visual vocabulary. If the new keyframe is found in the visual vocabulary, the loop closing module optimizes the entire map using pose graph optimization and global BA. Otherwise, the visual vocabulary of the new keyframe is added to the place recognition database.

As ORB-SLAM2 does not provide a method to save and load the map into a file, we implemented the method to save and load the map. The visual sparse map generated by ORB-SLAM2 contains visual feature points, keyframes and a pose graph. Each visual feature point has the index and 3D position in the map. Each keyframe has the index, 3D pose and visual feature descriptors. The pose graph represents connectivity among keyframes using vertices and edges. In the pose graph, vertices represent keyframes and edges represent visible connection among keyframes.

B. Localization

Given the map, only the tracking module is used in the localization mode. The local map and the map database is not updated in the localization mode. In addition, the place recognition database is not updated. Whenever the new image is captured, the tracking module computes the relative pose of the camera with respect to the origin of the map. The camera pose $\boldsymbol{x}_C$ is composed of the camera position $[x, y, z]$ and orientation $[roll, pitch, yaw]$ in the map. The coordinate of the map locates at the pose of the first keyframe in the map.

IV. Waypoint Follower

Using the camera pose and a reference line from the visual sparse map, the waypoint follower module computes the translation velocity and the angular velocity to control the robot. We assume $\boldsymbol{x}_C$ is identical to the robot pose $\boldsymbol{x}_R$ because the reference line is generated with assuming $\boldsymbol{x}_C$ is identical to $\boldsymbol{x}_R$. When a new image is captured, $\boldsymbol{x}_R$ is computed by the tracking module of ORB-SLAM2.

The reference line is generated from the map. The reference line is represented as the list of the keyframe positions

$$\boldsymbol{L}_R = \{\boldsymbol{x}_0, \boldsymbol{x}_1 \dots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_k\} \quad (1)$$

where $\boldsymbol{x}_k = [x, y, z]$ is the k^{th} keyframe position in the map.

If $\boldsymbol{x}_R$ is successfully computed by the tracking module, the nearest keyframe $\boldsymbol{x}_N$ from $\boldsymbol{x}_R$ is founded in $\boldsymbol{L}_R$. A keyframe ahead with a pre-defined distance from $\boldsymbol{x}_N$ is determined as a temporary target waypoint $\boldsymbol{x}_T$. Transitional difference $\boldsymbol{t}_D$ and angular difference θ_D between $\boldsymbol{x}_R$ and $\boldsymbol{x}_T$ can be computed by

$$\boldsymbol{t}_D = \|\boldsymbol{t}_T - \boldsymbol{t}_R\| \quad (2)$$

$$\theta_D = |\theta_T - \theta_R| \quad (3)$$

Where $\boldsymbol{t}_T = [x, y, z]$ and $\boldsymbol{t}_R = [x, y, z]$ are robot positions at the target waypoint and the current position respectively. θ_T and θ_R are orientations of the robot at target waypoint and current position respectively in 2D space.

To control the robot, we computes the translational velocity $\boldsymbol{V}_T$ and the rotational velocity $\boldsymbol{V}_\theta$ by

$$\boldsymbol{V}_T = \begin{cases} V_m & \text{if } \theta_D \leq \theta_h \\ \frac{V_m}{2} & \text{otherwise} \end{cases} \quad (4)$$

$$\boldsymbol{V}_\theta = \frac{\theta_D}{\alpha} \quad (5)$$

where V_m is the desired maximum translational speed of the robot. θ_h is a threshold of angular difference for reducing $\boldsymbol{V}_T$. If θ_D is larger than θ_h, $\boldsymbol{V}_T$ is reduced by half. α is an empirical coefficient for computing $\boldsymbol{V}_\theta$ using θ_D.

V. Experimental Results

We evaluated the proposed autonomous navigation system using Robotis Turtlebot 2 [23] with Orbbec Astra Pro [24] in indoor environment and Clearpath Husky [25] with Logitech C920 Pro [26] in outdoor environment.

A. Experimental platforms

We installed one RGB-D camera, Orbbec Astra Pro, on the Turtlebot in indoor environment as shown in Fig. 3. Orbbec Astra Pro has a resolution of 640 × 480 pixels in both a color image and depth image.

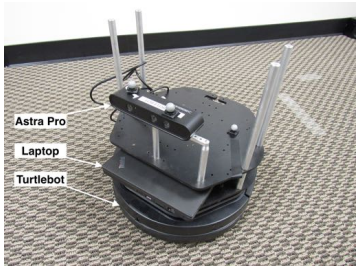


Figure 3. Robotis Turtlebot 2 with Orbbec Astra Pro for indoor environment

As the RGB-D camera is not working in outdoor environment, we use Logitech C920 Pro instead of Orbbec Astra Pro. We use only 640 × 480 color images for both mapping and localization in outdoor environment. In addition, we use Clearpath Husky for safe and robust mobility in outdoor environment as shown in Fig. 4. The autonomous navigation systems in both robot platforms are built on ROS [27].

Figure 4. Clearpath Husky with a Logitech C920 for outdoor environment

B. Localization accuracy with Map data

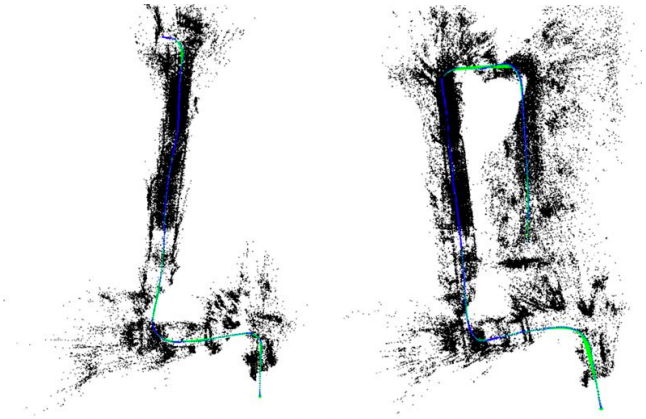
We evaluated localization accuracy with map data before evaluating autonomous navigation. We use the same map data for evaluating localization accuracy. However, we use only color images for localization while both color images and depth images are used for building a map in indoor environment.

Figure 5. Snapshots of offices and hallway for datasets in indoor environment. (a) office A, (b) hallway between office A and elevator, (c) elevator at the end of hallway, (d) glass door to office B, (e) narrow gate to office B and (f) office B.

We collected three datasets in office environment as shown in Fig. 5. The first dataset is collected in office A which includes desks, chairs and shelves. The robot starts near the first shelf and returns to the start position. The second dataset is collected in Office A and a hallway. The robot starts from Office A, runs along the hallway and stops in front of an elevator at the end of the hallway. The third dataset is collected in Office A, the hallway and Office B. The robot starts from Office A, runs along the hallway and stops at Office B. There is a 1 meter-wide narrow gate between the hallway and Office B. Table I shows the path length and environment of each dataset. Fig. 6 shows maps and trajectories of dataset II and III. The map and trajectory of Dataset I is shown in Fig. 2.

TABLE I. Datasets in Indoor Environment

Dataset	Length [m]	Environment
I	17.41	Office A
II	41.38	Office A, hallway
III	49.40	Office A and B, hallway



(a) Dataset II (b) Dataset III

Figure 6. Maps and trajectories in Dataset II and III

Table II shows the localization error with map datasets. Although the same map dataset is used for evaluating localization accuracy, the average Root Mean Square Error (RMSE) is 0.031 meter because ORB-SLAM2 randomly generates visual features from a color image for localization. However, the average RMSE is acceptable for autonomous navigation because the minimum width of path is 1 meter. Fig. 7 shows map and localization trajectories on dataset I. As RMSE is 0.036 meter, the localization trajectory overlays the map trajectory.

TABLE II. LOCALIZTION RMSE WITH MAP DATA

Dataset	RMSE [m]
I	0.036
II	0.03
III	0.03
Average	0.031

We also evaluated localization accuracy in environment changes because the environment can be changed after generating the map. We changed about 30% of objects in the same place in dataset I and collected a new dataset for evaluating localization. Given the map generated from dataset I, localization RMSE is 0.116 ± 0.111 meter [mean ± standard deviation]. Although environment changes increase localization RMSE slightly, the RMSE in environment changes is still acceptable for autonomous navigation.

TABLE III. LOCALIZTION RMSE IN AUTONOMOUS NAVIGATION

Dataset	RMSE [m]
I	0.065 ± 0.045
II	0.166 ± 0.127
III	0.117 ± 0.075
Average	0.116 ± 0.082

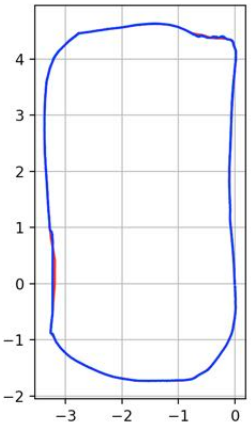
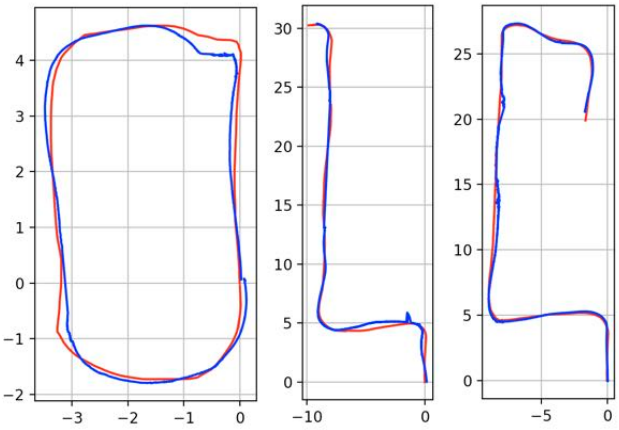


Figure 7. Map and localization trajectories with Dataset I. Red line represents the map trajectory and blue line represents the localization trajectory.

C. Localization accuracy in autonomous navigation

We evaluated localization error when the robot runs in the autonomous navigation phase. The waypoint follower enables the robot to follow a reference line as close as possible. We compute the localization error by finding the closest waypoint from the estimated position by ORB-SLAM2 localization as shown in Table III.



(a) Dataset I (b) Dataset II (c) Dataset III

Figure 8. Map and localization trajectories in autonomous navigation. Red lines represent trajectories of maps and bule lines represent trajectories of localization.

Experimental results show that: 1) the average localization RMSE is 0.116 ± 0.082 meter [mean ± standard deviation]; 2) the robot successfully navigates in three different environments even there are challenge environments such as a feature-spare long hallway (length: 25 meter) and the 1 meter-wide narrow gate; 3) there are relatively larger error when the robot turns; 4) the feature sparse long hallway increases localization error. Fig. 8 shows map and localization trajectories in autonomous navigation.

D. Environment changes in outdoor environment

We evaluated localization error with environment changes in outdoor environment. Datasets are collected along the

sidewalk around JD.com office, Santa Clara, California, USA. The path consists of straight, curved and winding sidewalks under trees as shown in Fig. 9.

(a) (b) (c) (d)

Figure 9. Snapshots of outdoor environment. (a) start position, (b) curved sidewalk, (c) winding sidewalk and (d) goal position.

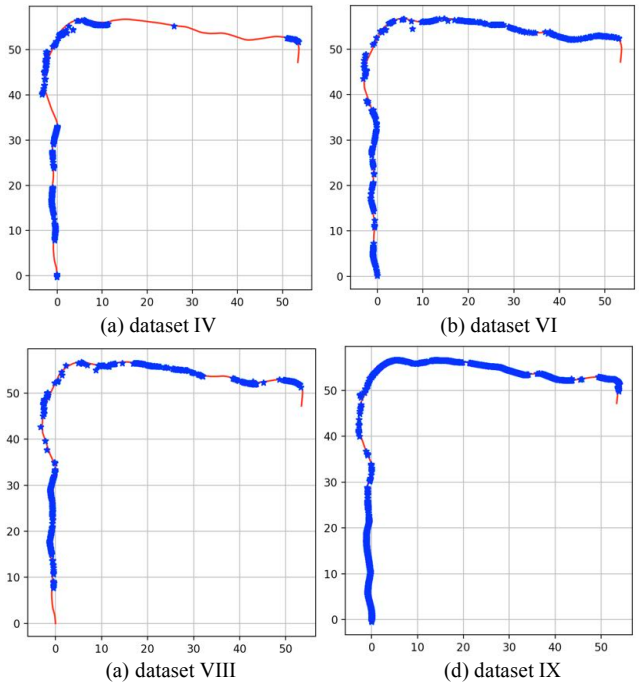
The map dataset is collected at 15:04 on December 13, 2017. The path length of the map is 114.70 meter. We collected six datasets as shown in Table IV: 1) dataset IV to VII are collected at different time in sunny days; 2) dataset VIII is collected in a cloudy day; 3) dataset IX is collected in a rainy day.

TABLE IV. LOCALIZATION ANAYSIS WITH ENVIRONMENT CHANGES IN OUTDOOR ENVIRONMENT

Dataset	Weather	Date/Time	Failure ratio	Failure time [sec]		
				Max	Mean	Std.
IV	Sunny	2018-01-19-09-57-51	48%	36.15	1.55	4.29
V	Sunny	2018-01-11-14-12-09	10%	0.57	0.22	0.13
VI	Sunny	2018-01-12-15-32-45	3%	0.33	0.07	0.06
VII	Sunny	2018-01-12-16-51-56	12%	2.40	0.44	0.52
VIII	Cloudy	2018-01-17-11-39-49	17%	3.43	0.99	1.30
IX	Rainy	2018-01-03-11-40-42	12%	9.80	0.55	1.30

We use two metric, failure ratio and failure time, for evaluating localization performance. Failure ratio is the ratio of localization failure over all localization tries. Failure time is the time from the localization failure to the next localization success. As the dataset is collected by manual driving, localization accuracy is not evaluated.

As shown in Table IV, experimental results show that: 1) dataset VI has the smallest failure ratio because dataset VI is collected at similar time and weather to the map; 2) dataset IV has the largest failure ratio because the illumination of dataset IV is quite different from the map due to the position of the sun; 3) failure time has proportional relationship with failure ratio in sunny day but the proportional relationship between failure ratio and failure time is not valid in the rainy day and the cloudy day; 4) in the rainy day, failure time is larger than the cloudy day while failure ratio is smaller than the cloudy day. Fig. 10 shows trajectories of map and localization in dataset IV, VI, VIII and IX.



(a) dataset IV (b) dataset VI

(a) dataset VIII (d) dataset IX

Figure 10. Map and localization trajectories with environment changes in outdoor environment. Red lines represent trajectories of maps and bule stars represent positions of successful localization.

E. Autonomous navigation in outdoor environment

As mentioned in the previous section, ORB-SLAM2 is not robust at different time and different weather in outdoor environment. Thus, we evaluated autonomous navigation at 15:02 on January 11, 2018, a sunny day, which is similar time and weather to the map.

Experimental result shows the robot ran successfully on the sidewalk and localization RMSE is 0.246 ± 0.151 meter [mean ± standard deviation]. The width of sidewalk is about 1.5 meter. Fig. 11 shows trajectories of map and localization in autonomous navigation. We note that the robot is rarely localized in the curved sidewalk because most visual features come from the distant objects.

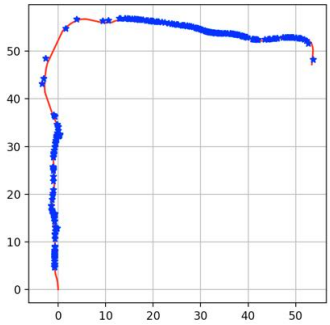


Figure 11. Map and localization trajectories in autonomous navigation in outdoor environment. Red line represents trajectories of map and bule stars represents positions of successful localization.

VI. CONCLUSION

We proposed an autonomous navigation system using only

visual sparse map for indoor environment and outdoor environment. ORB-SLAM2 is used for mapping and localization. Waypoint follower enables the robot to follow the reference line. We evaluated the proposed system in indoor environment and outdoor environment using two robot platforms.

Experimental results show that: 1) localization errors with the map datasets are acceptable for the robot to run autonomously indoor environment; 2) the robot successfully ran in three indoor environments including environment changes; 3) environment changes in outdoor apparently increases localization failure ratios; 4) the robot successfully ran in similar time and weather to the map in outdoor environment.

We will investigate for robust localization with environment changes in outdoor environment. In addition, sensor fusion with additional sensors such as IMU, GPS and Lidar will be investigated. We will also extend the proposed system by including obstacle avoidance and path planning.

ACKNOWLEDGMENT

We would like to thank Chengzhi Qi and Jiawei Yao for collecting datasets with different time and weather conditions.

REFERENCES

[1] R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza, "Introduction to Autonomous Mobile Robots, Second edition," The MIT Press, 2011.

[2] J.C. Latombe, "Robot Motion Planning," Kluwer Academic Publishers, 1991.

[3] A. Lazanas, J.C. Latombe, "Landmark robot navigation," Proceedings of the Tenth National Conference on AI, 1992.

[4] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," In Proc. IEEE International Conference on Robotics and Automation (ICRA), pages 300-307, 2010.

[5] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," Exp. Robot., pp. 179190, 2008.

[6] F. Dayoub, T. Morris, B. Upcroft, P. Corke, "Vision- only autonomous navigation using topometric maps," Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on , vol., no., pp.1923,1929, 3-7 Nov. 2013.

[7] J. Engel, T. Schps, D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," In European Conference on Computer Vision (ECCV), 2014

[8] J. Engel, J. Sturm, D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," In IEEE International Conference on Computer Vision (ICCV), 2013

[9] S. Hong, "6-DOF Pose Estimation for A Portable Navigation Device," PhD dissertation, University Of Arkansas at Little Rock, 2014.

[10] R.F. Salas-Moreno, R.A. Newcombe, H. Strasdat, P.H.J. Kelly, A.J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1352-1359, 2013.

[11] R.F. Salas-Moreno, B. Glocken, P.H.J. Kelly, A.J. Davison, "Dense planar SLAM," IEEE International Symposium on Mixed and Augmented Reality (IS- MAR), pp.157-164, 2014.

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping Using Kinect-style depth cameras for dense 3D modeling of indoor environments," International Journal of Robotics Research, vol. 31, no. 5, pp. 647663, 2012.

[13] A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Mat- urana, D. Fox, N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," In International Symposium on Robotics Research (ISRR), pp. 1-16, 2011.

[14] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohi, J. Shotton, S. Hodges, A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," IEEE International Symposium on Mixed and Augmented Reality (ISMAR), , pp.127-136, 2011

[15] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, S. Bathiche, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on , pp.83,88, 2013.

[16] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052-1067, 2007.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB- SLAM : a Versatile and Accurate Monocular," IEEE Trans. Robot., pp. 115, 2015.

[18] H. Lim, J. Lim, and H. J. Kim, "Real-time 6-DOF monocular visual SLAM in a large-scale environment," IEEE Inter- national Conference on Robotics and Automation (ICRA), 2014, pp. 1532-1539, 2014.

[19] J. Lim, J.-M. Frahm, and M. Pollefeys, "Online environment mapping," CVPR, pp. 3489-3496, 2011.

[20] H. Badino, D. Huber, and T. Kanade, "Real-time topometric localization," IEEE Int. Conf. Robot. Autom., no. ICRA, pp.1635-1642, May 2012.

[21] C. X. Guo, K. Sartipi, R. C. Dutoit, G. Georgiou, R. Li, J. O. Leary, E. D. Nerurkar, J. A. Hesch, and S. I. Roumeliotis, "Large-Scale Cooperative 3D Visual-Inertial Mapping in a Manhattan World," Technical Report, University of Minnesota, Dept. of Comp. Sci. and Eng., MARS Lab, February 2015.

[22] R. Mur-Artal, & J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," IEEE Transactions on Robotics, 2017.

[23] Robotis turtlebot2, http://www.turtlebot.com/turtlebot2/

[24] Orbbec Astra Pro, https://orbbec3d.com/product-astra-pro/

[25] Clearpath Husky, https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[26] Logitech C920 Pro, https://www.logitech.com/en-us/product/hd-pro-webcam-c920

[27] Robot Operating System (ROS), http://www.ros.org/

[28] Vicon motion capture system, https://www.vicon.com/

[29] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, "Outdoor autonomous navigation using monocular vision," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.

Socially Invisible Navigation for Intelligent Vehicles

Aniket Bera[1], Tanmay Randhavane[1], Emily Kubin[2], Austin Wang[1], Kurt Gray[2], and Dinesh Manocha[1]

***Abstract*— We present a real-time, data-driven algorithm to enhance the *social-invisibility* of autonomous vehicles within crowds. Our approach is based on prior psychological research, which reveals that people notice and–importantly–react negatively to groups of social actors when they have high *entitativity*, moving in a tight group with similar appearances and trajectories. In order to evaluate that behavior, we performed a user study to develop navigational algorithms that minimize entitativity. This study establishes mapping between emotional reactions and multi-robot trajectories and appearances, and further generalizes the finding across various environmental conditions. We demonstrate the applicability of our entitativity modeling for trajectory computation for active surveillance and dynamic intervention in simulated robot-human interaction scenarios. Our approach empirically shows that various levels of entitative robots can be used to both avoid and influence pedestrians while not eliciting strong emotional reactions, giving multi-robot systems socially-invisibility.**

I. Introduction

As robots have become more common in social environments, people's expectations of their social skills have increased. People often want robots to be more socially visible–more salient social agents within group contexts [17]. This social visibility includes being more capable of drawing the attention of humans and evoking powerful emotions [21]. Cases of social visibility include tasks in which robots must work collaboratively with humans. However, not all contexts require socially visible robots. There are situations in which robots are not used to collaborate with people but instead used to monitor them. In these cases, it may be better for robots to be socially invisible.

Social invisibility refers to the ability of agents to escape the attention of other people. For example, psychological research reveals that African Americans often go unnoticed in social environments[10], especially reactions related to threat. Evolution has attuned the human brain to respond rapidly to threatening stimuli, thus the less a person–or a robot–induces negative emotion, the less likely it is to be noticed within a social milieu. The social invisibility conferred by not inducing emotion is especially important in surveillance contexts in which robots are expected to move seamlessly among people without being noticed. Noticing surveillance robots not only makes people hide their behavior, but the negative emotions that prompt detection may also induce reactance [8], which may lead to people to lash out and harm the robots or even other people [11] Research reveals a number of ways of decreasing negative emotional reactions towards social agents [9], but one element may be especially important for multi-robot systems: entitativity [12], "groupiness") is tied to three main elements, uniformity of appearance, common movement, and proximity to one another. The more agents look and move the same, and the closer agents are to each other, the more entitative a group seems, which is why a marching military platoon seems more grouplike than people milling around a shopping mall.

The threatening nature of groups means that the more entitative (or grouplike) a collection of agents seem, the greater the emotional reaction they induce and the greater their social visibility. As maximizing the social invisibility of collections of agents requires minimizing perceptions of threat, it is important for multi-robot systems to minimize their entitativity. In other words, if multi-robots systems are to move through groups without eliciting negative reactions [16], they must seem more like individuals and less like a cohesive and coordinated group.

Main Results: We present a novel, real-time planning algorithm that seeks to optimize entitativity within pedestrian environments in order to increase socially-invisible navigation (by minimizing negative emotional reactions). First, we conduct a user study to empirically tie trajectories of multi-robot systems to emotional reactions, revealing that–as predicted–more entitative robots are seen as more unnerving. Second, we generalize these results across a number of different environmental conditions (like lighting). Third, we extract the trajectory of each pedestrian from the video and use Bayesian learning algorithms to compute their motion model. Using entitativity features of groups of robots and the pedestrians, we perform long-term path prediction for the pedestrians. To determine these entitativity features we establish a data-driven entitativity mapping (EDM) between the group robot motion and entitativity measure from an elaborate web-based perception user study that compares the participants' emotional reactions towards simulated videos of multiple robots. Specifically, highly entitative collections of robots are reported as unnerving and uncomfortable. The results of our mapping are well supported by psychology literature on entitativity [33]. Our approach is an extension of the approach presented in [5]. We refer the readers to read [5] for the technical details.

We highlight the benefits of our data-driven metric for use of multiple robots for crowd surveillance and active interference. We attempt to provide maximally efficient navigation and result in maximum social invisibility. In order to pursue different sets of scenarios and applications, we highlight the

[1]Authors from the Department of Computer Science, University of North Carolina at Chapel Hill, USA

[2]Authors from the Department of Psychology and Neuroscience, University of North Carolina at Chapel Hill, USA

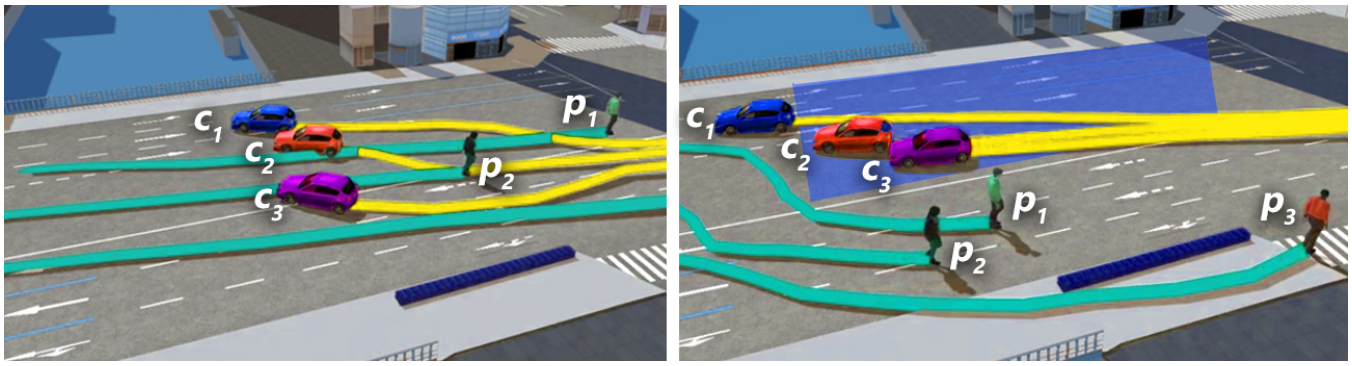


Fig. 1: *Multi-robot navigation systems (vehicles (c_x) marked by yellow trajectories) navigate amongst crowds. Our novel navigation algorithm takes into account various levels of physical and social constraints and use them for: (a) Active Navigation in the presence of pedestrians (teal trajectories) while moving through them with no collisions; (b) Dynamic intervention where the robots try to influence the crowd behavior and movements and make the pedestrians avoid the area marked by a dark blue overlay.*

performance of our work in multiple surveillance scenarios based on the level of increasing social interaction between the robots and the humans.

Our approach has the following benefits:

1. Entitativity Computation: Our algorithm accurately predicts emotional reactions (entitativity) of pedestrians towards robots in groups.

2. Robust computation: Our algorithm is robust and can account for noise in pedestrian trajectories, extracted from videos.

3. Fast and Accurate: Our algorithm involves no precomputation and evaluates the entitativity behaviors at interactive rates.

The rest of the paper is organized as follows. In Section 2, we review the related work in the field of psychology and behavior modeling. In Section 3, we give a background on quantifying entitativity and introduce our notation. In Section 4, we present our interactive algorithm, which computes the perceived group entitativity from trajectories extracted from video. In Section 5, we describe our user study on the perception of multiple simulated robots with varying degrees of entitativity.

II. Related Work

Human beings are inherently social creatures, making interacting with and perceiving others an important part of the human experience. Complex interactions within brain regions work harmoniously to navigate the social landscape [36]. Interesting patterns emerge when attempting to understand how humans view groups of people.

A. Psychological Perspectives on Group Dynamics

A long-standing tenet of social psychology is that people's behaviors hinge upon their group context. Importantly, the impact of social dynamics is highly influenced by group contexts [39]–often for the worse. Decades of psychological research reveals that people interact more negatively with groups than with individuals [33], expressing more hostility towards and feeling more threatened by a group than an individual [16].

B. Human-Aware Robot Navigation

Many approaches have been applied towards the navigation of socially-aware robots [30], [25], [29], [19], [26], [24], [7], [41], [32]. This type of navigation can be generated by predicting the movements of pedestrians and their interactions with robots [26]. Some algorithms use probabilistic models in which robots and human agents cooperate to avoid collisions [40]. Other techniques apply learning models which have proven useful in adapting paths to social conventions [27], [31], [34], [14]. Yet other methods model personal space in order to provide human-awareness [1]. This is one of many explicit models for social constraints [38], [23], [13]. While these works are substantial, they do not consider psychological constraints or pedestrian personalities.

C. Behavior Modeling of Pedestrians

There is considerable literature in psychology, robotics, and autonomous driving on modeling the behavior of pedestrians. Many rule-based methods have been proposed to model complex behaviors based on motor, perceptual, behavioral, and cognitive components [37], [15]. There is extensive literature focused on modeling emergent behaviors [35]. Other techniques have been proposed to model heterogeneous crowd behaviors based on personality traits [6], [2], [22], [3].

III. Social Interaction

In this section, we present our interactive algorithm for performing socially-invisible robot navigation in crowds. Our approach can be combined with almost any real-time pedestrian tracker that works on dense crowd videos. Figure

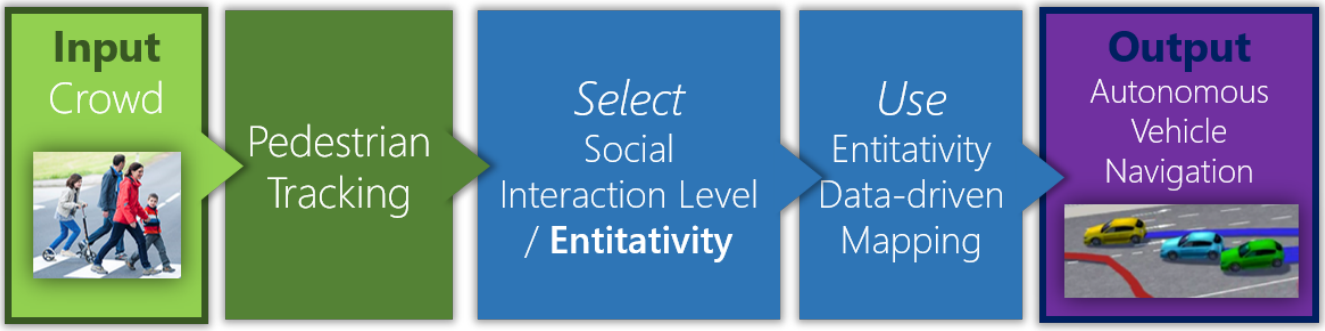


Fig. 2: *Our method takes a live or streaming crowd video as an input. We extract the initial set of pedestrian trajectories using an online pedestrian tracker. Based on the level of social invisibility we want to achieve, we compute motion model parameters of the robot group using a data-driven entitativity mapping (which we compute based on a user-study(Section IV)).*

2 gives an overview of our approach. Our method takes a live or streaming crowd video as an input. We extract the initial set of pedestrian trajectories using an online pedestrian tracker. Based on the level of social invisibility we want to achieve, we find motion model parameters of the robot group using a data-driven entitativity mapping (which we compute based on a user-study(Section IV)).

A. Entitativity

Entitativity is the perception of a group comprised of individuals as a single entity. People sort others into entities like they group together objects in the world, specifically by assessing common fate, similarity, and proximity [12]. When individuals are connected by these properties, we are more likely to perceive them as a single entity. Larger groups are more likely to be perceived as entities, but only when there is similarity among the groups individual members [28].

Entitativity is the extent to which a group resembles a single entity versus of collection of individuals; in other words, it is the groups "groupiness" or "tightness" [12], [20]. Overall, entitativity is driven by the perception of three main elements:

1. Uniformity of appearance: Highly entitative groups have members that look the same.

2. Common movement: Highly entitative groups have members that move similarly.

3. Proximity: Highly entitative groups have members that are very close to each other.

B. Notation and Terminology

The motion model is the local navigation rule or scheme that each agent uses to avoid collisions with other agents or obstacles and has a group strategy. The parameters of the motion model is denoted $\mathbf{P} \in \mathbb{R}^6$. We based our model on the RVO velocity-based motion model [42]. In this model, the motion of each agent is governed by these five individual pedestrian characteristics: *Neighbor Dist, Maximum Neighbors, Planning Horizon, (Radius) Personal Space, and Preferred Speed* and one group characteristic: *Group Cohesion*. We combine RVO with a group navigation scheme in Section 4.2. In our approach, we mainly analyze four parameters ($\mathbf{GP} \in \mathbb{R}^4$): *Neighbor Dist, (Radius) Personal Space, Group Cohesion, and Preferred Speed*.

Trajectories extracted from real-world scenarios are likely to have incomplete tracks and noise [18]. Therefore, the state of each agent is computed using a Bayesian inference technique in order to compensate for such errors.

Entitativity Metric: Prior research in psychology takes into account properties such as uniformity, common movement, and proximity, and models the perception of *entitativity* using the following 4-D feature vector:

$$\mathbf{E} = \begin{pmatrix} Friendliness \\ Creepiness \\ Comfort \\ Unnerving \end{pmatrix} \quad (1)$$

Friendliness, Creepiness, Comfort and *Unnerving (ability to unnerve)* are the emotional impressions made by the group on observers. Using Cronbach's α (a test of statistical reliability) in pilot studies we observed that the parameters were highly related with $\alpha = 0.794$, suggesting that they were justifiable adjectives for socially-invisible navigation.

IV. Data-Driven Entitativity Model

We performed a user study to understand the perception of multiple pedestrians and vehicles with varying degrees of entitativity. For the details of the user study, we refer the readers to read [5].

Given the entitativity features obtained using the psychology study for each variation of the motion model parameters, we can fit a generalized linear model to the entitativity features and the model parameters. We refer to this model as the *Data-Driven Entitativity Model*. For each video pair i in the gait dataset, we have a vector of parameter values and a

vector of entitativity features $\mathbf{E}_i$. Given these parameters and features, we compute the entitativity mapping of the form:

$$\begin{pmatrix} Friendliness \\ Creepiness \\ Comfort \\ Unnerving \end{pmatrix} = \mathbf{G_{mat}} * \begin{pmatrix} \frac{1}{14}(Neighbor\ Dist - 5) \\ \frac{1}{3.4}(Radius - 0.7) \\ \frac{1}{2}(Pref.\ Speed - 1.5) \\ \frac{1}{1.8}(Group\ Cohesion - 0.5) \end{pmatrix} \tag{2}$$

We fit the matrix $\mathbf{G_{mat}}$ using generalized linear regression with each of the entitativity features as the responses and the parameter values as the predictors using the normal distribution:

$$\mathbf{G_{mat}} = \begin{pmatrix} -1.7862 & -1.0614 & -2.1983 & -1.7122 \\ 1.1224 & 1.1441 & 1.7672 & -0.2634 \\ -1.0500 & -1.2176 & -2.1466 & -0.9220 \\ 1.1948 & 1.7000 & 0.9224 & 0.3622 \end{pmatrix}. \tag{3}$$

We can make many inferences from the values of $\mathbf{G_{mat}}$. The negative values in the first and third rows indicate that as the values of motion model parameters increase, the friendliness of the group decreases. That is, fast approaching and cohesive groups appear to be less friendly. This validates the psychological findings in previous literature. One interesting thing to note is that creepiness increases when group cohesion decreases. When agents/pedestrians walk in a less cohesive group, they appear more creepy but they may appear less unnerving.

We can use our data-driven entitativity model to predict perceived entitativity of any group for any new input video. Given the motion parameter values $\mathbf{GP}$ for the group, the perceived entitativity or group emotion $\mathbf{GE}$ can be obtained as:

$$\mathbf{GE} = \mathbf{G_{mat}} * \mathbf{GP} \tag{4}$$

A. Socially-Invisible Vehicle Navigation

To provide socially-invisible navigation, we use the entitativity level of robots. We control the entitativity level depending on the requirements of the social-invisibility. We represent the social-invisibility as a scalar $s \in [0, 1]$ with $s = 0$ representing very low social-invisibility and $s = 1$ representing highly socially-invisible robots. Depending on the applications and situations, the social-invisibility can be varied.

We relate the desired social-invisibility (s) to entitativity features $\mathbf{GE}$ as follows:

$$s = 1 - \frac{\|\mathbf{GE} - \mathbf{GE}_{min}\|}{\|\mathbf{GE}_{max} - \mathbf{GE}_{min}\|} \tag{5}$$

where $\mathbf{GE}_{max}$ and $\mathbf{GE}_{min}$ are the maximum and minimum entitativity values obtained from the psychology study.

According to Equation 5, there are multiple entitativity features $\mathbf{GE}$ for the desired social-invisibility s. This provides flexibility to choose which features of entitativity we wish to adjust and we can set the desired entitativity $\mathbf{GE}_{des}$ that provides the desired social-invisibility level. Since $\mathbf{G_{mat}}$ is invertible, we can compute the motion model parameters $\mathbf{GP}_{des}$ that achieve the desired entitativity:

$$\mathbf{GP}_{des} = \mathbf{G_{mat}}^{-1} * \mathbf{E}_{des} \tag{6}$$

These motion model parameters $\mathbf{GP}_{des}$ are the key to enabling socially-invisible collision-free robot navigation through a crowd of pedestrians. Our navigation method is based on Generalized Velocity Obstacles (GVO) [43], which uses a combination of local and global methods. The global metric is based on a roadmap of the environment. The local method computes a new velocity for each robot and takes these distances into account. Moreover, we also take into account the dynamic constraints of the robot in this formulation - for example, mechanical constraints that prevent the robot from rotating on the spot.

V. Applications

We present some driving applications of our work that are based on use of multiple autonomous car navigation systems. In these scenarios, our method optimizes multi-robot systems so that they can interact with such crowds seamlessly based on physical constraints (e.g. collision avoidance, robot dynamics) and social invisibility. We simulate our algorithm with two sets of navigation scenarios based on the level of increasing social interaction between the robots and the humans:

1) Active Navigation: This form of navigation includes autonomous robots that share a physical space with pedestrians. While performing navigation and analysis, these robots will need to plan and navigate in a collision-free manner in real-time amongst crowds. In this case, the robots need to predict the behavior and trajectory of each pedestrian. For example, marathon races tend to have large populations, with a crowd whose location is constantly changing. In these scenarios, it is necessary to have a navigation system that can detect shifting focal points and adjust accordingly.

In such scenarios, the robots need to be highly socially-invisible ($s = 0$). We achieve this by setting the entitativity features to the minimum $\mathbf{E} = \mathbf{E}_{min}$ (Equation 5).

2) Dynamic intervention: In certain scenarios, robots will not only share a physical space with people but also influence pedestrians to change or follow a certain path or behavior. Such interventions can either be overt, such as forcing people to change their paths using visual cues or pushing, or subtle (for example, nudging). This type of navigation can be used in any scenario with highly dense crowds, such as a festival or marathon. High crowd density in these events can lead to stampedes, which can be very deadly. In such a scenario, a robot can detect when density has reached dangerous levels and intervene, or "nudge" individuals until they are distributed more safely.

For dynamic intervention with pedestrians or robots, we manually vary the entitativity level depending on urgency or agent proximity to the restricted area. In these situations,

we restrict the entitativity space by imposing a lower bound s_{min} on the social-invisibility (Equation 5):

$$s_{min} \leq 1 - \frac{\|\mathbf{E} - \mathbf{E}_{min}\|}{\|\mathbf{E}_{max} - \mathbf{E}_{min}\|}. \quad (7)$$

A. Performance Evaluation

We evaluate the performance of our socially-invisible navigation algorithm with GVO [43], which by itself does not take into account any social constraints. We compute the number of times a pedestrian intrudes on a designated restricted space, and thereby results in issues related to navigating through a group of pedestrians. We also measure the additional time that a robot with our algorithm takes to reach its goal position, without the pedestrians intruding a predesignated restricted area. Our results (Table I) demonstrate that in $< 30\%$ additional time, robots using our navigation algorithm can reach their goals while ensuring that the restricted space is not intruded. Table I also lists the time taken to compute new trajectories while maintaining social invisibility. We have implemented our system on a Windows 10 desktop PC with Intel Xeon E5-1620 v3 with 16 GB of memory.

Dataset	*Additional Time*	*Intrusions Avoided*	*Performance*
NPLC-1	14%	3	3.00E-04 ms
NDLS-2	13%	2	2.74E-04 ms
IITF-1	11%	3	0.72E-04 ms
NDLS-2	17%	4	0.98E-04 ms
NPLC-3	14%	3	1.27E-04 ms
NDLS-4	13%	2	3.31E-04 ms
IITF-2	11%	3	1.76E-03 ms

TABLE I: ***Navigation Performance for Dynamic Intervention:*** *A robot using our navigation algorithm can reach its goal position, while ensuring that any pedestrian does not intrude the restricted space with $< 15\%$ overhead. We evaluated this performance in a simulated environment, though the pedestrian trajectories were extracted from the original dataset [4]. In all the videos we have manually annotated a specific area as the restricted space.*

VI. Conclusions, Limitations and Future Work

Drawing from work in social psychology, we develop a novel algorithm to minimize entitativity and thus maximize the social invisibility of multi-robot systems within pedestrian crowds. A user-study confirms that different entitativity profiles–as given by appearance, trajectory and spatial distance–are tied to different emotional reactions, with high entitativity groups evoking negative emotions in participants. We then use trajectory information from low-entitative groups to develop a real-time navigation algorithm that should enhance social invisibility for multi-robot systems.

Our approach has some limitations. Although we did generalize across a number of environmental contexts, we note that motion-based entitativity is not the only feature involved in social salience and other judgments. People use a rich set of cues when forming impressions and emotionally reacting to social agents, including perceptions of race, class, religion, and gender. As our algorithm only uses motion trajectories, it does not exhaustively capture all relevant social features. However, motion trajectories are an important low-level feature of entitativity and one that applies especially to robots, who may lack these higher-level social characteristics.

Future research should extend this algorithm to model the appearances of robots in multi-robot systems. Although many social cues may not be relevant to robots (e.g., race), the appearance of robots can be manipulated. Research suggests that robots that march will have higher entitativity and hence more social visibility. This may prove a challenge to manufacturers of autonomous vehicles, as mass production typically leads to identical appearances. Another key future direction involves examining the interaction of the perceiver's personality with the characteristics of multi-robot systems, as some people may be less likely to react negatively to entitative groups of robots, perhaps because they are less sensitive to general threat cues or, more specifically, have more experience with robots.

VII. Acknowledgements

This research is supported in part by ARO grant W911NF16-1-0085, and Intel.

References

[1] Marie-Lou Barnaud et al. Proxemics models for human-aware navigation in robotics: Grounding interaction and personal space models in experimental data from psychology. In *Proceedings of the 3rd IROS2014 workshop Assistance and Service Robotics in a Human Environment*, 2014.

[2] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Interactive crowd-behavior learning for surveillance and training. *IEEE Computer Graphics and Applications*, 36(6):37–45, 2016.

[3] Aniket Bera, Sujeong Kim, and Dinesh Manocha. Modeling trajectory-level behaviors using time varying pedestrian movement dynamics. *Collective Dynamics*, 3:1–23, 2018.

[4] Aniket Bera and Dinesh Manocha. Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4164–4169. IEEE, 2014.

[5] Aniket Bera, Tanmay Randhavane, Emily Kubin, Austin Wang, Kurt Gray, and Dinesh Manocha. The socially invisible robot: Navigation in the social world using robot entitativity. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*, 2018.

[6] Aniket Bera, Tanmay Randhavane, and Dinesh Manocha. Aggressive, tense, or shy? identifying personality traits from crowd videos. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 112–118, 2017.

[7] Aniket Bera, Tanmay Randhavane, Rohan Prinja, and Dinesh Manocha. Sociosense: Robot navigation amongst pedestrians with social and psychological constraints. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 7018–7025, 2017.

[8] Jack W Brehm. A theory of psychological reactance. 1966.

[9] Lisanne Brown, Kate Macintyre, and Lea Trujillo. Interventions to reduce hiv/aids stigma: what have we learned? *AIDS education and prevention*, 15(1):49–69, 2003.

[10] Jazmin L Brown-Iannuzzi, Kelly M Hoffman, B Keith Payne, and Sophie Trawalter. The invisible man: Interpersonal goals moderate inattentional blindness to african americans. *Journal of Experimental Psychology: General*, 143(1):33, 2014.

[11] Brad J Bushman, Angelica M Bonacci, Mirjam Van Dijk, and Roy F Baumeister. Narcissism, sexual refusal, and aggression: Testing a narcissistic reactance model of sexual coercion. *Journal of Personality and Social Psychology*, 84(5):1027, 2003.

[12] Donald T Campbell. Common fate, similarity, and other indices of the status of aggregates of persons as social entities. *Systems research and behavioral science*, 3(1):14–25, 1958.

[13] Ernest Cheung, Aniket Bera, Emily Kubin, Kurt Gray, and Dinesh Manocha. Identifying driver behaviors using trajectory features for vehicle navigation. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*, 2018.

[14] Ernest Cheung, Aniket Bera, and Dinesh Manocha. Efficient and safe vehicle navigation based on driver behavior classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[15] Ernest Cheung, Tsan Kwong Wong, Aniket Bera, Xiaogang Wang, and Dinesh Manocha. Lcrowdv: Generating labeled videos for simulation-based crowd behavior learning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2016.

[16] Erin Cooley and B Keith Payne. Using groups to measure intergroup prejudice. *Personality and Social Psychology Bulletin*, 43(1):46–59, 2017.

[17] Brian R Duffy. Anthropomorphism and the social robot. *Robotics and autonomous systems*, 42(3-4):177–190, 2003.

[18] Markus Enzweiler and Dariu M Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE PAMI*, 2009.

[19] Gonzalo Ferrer, Anais Garrell, and Alberto Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *Intelligent robots and systems (IROS), 2013 IEEE/RSJ international conference on*, pages 1688–1694. IEEE, 2013.

[20] Wai-man Ip, Chi-yue Chiu, Ching Wan, et al. Birds of a feather and birds flocking together: Physical versus behavioral cues may lead to trait-versus goal-based group perception. *Journal of personality and social psychology*, 90(3):368, 2006.

[21] Sara Kiesler and Jennifer Goetz. Mental models of robotic assistants. In *CHI'02 extended abstracts on Human Factors in Computing Systems*, pages 576–577. ACM, 2002.

[22] Sujeong Kim, Aniket Bera, Andrew Best, Rohan Chabra, and Dinesh Manocha. Interactive and adaptive data-driven crowd simulation. In *Virtual Reality (VR)*, pages 29–38. IEEE, 2016.

[23] Rachel Kirby, Reid Simmons, and Jodi Forlizzi. Companion: A constraint-optimizing method for person-acceptable navigation. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 607–612. IEEE, 2009.

[24] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[25] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[26] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*, 2012.

[27] Matthias Luber, Luciano Spinello, Jens Silva, and Kai O Arras. Socially-aware robot navigation: A learning approach. In *Intelligent robots and systems (IROS), 2012 IEEE/RSJ international conference on*, pages 902–907. IEEE, 2012.

[28] Craig McGarty, S Alexander Haslam, Karen J Hutchinson, and Diana M Grace. Determinants of perceived consistency: The relationship between group entitativity and the meaningfulness of categories. *British Journal of Social Psychology*, 34(3):237–256, 1995.

[29] Billy Okal and Kai O Arras. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2889–2895. IEEE, 2016.

[30] Amit Kumar Pandey and Rachid Alami. A framework towards a socially aware mobile robot motion in human-centered dynamic environment. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5855–5860. IEEE, 2010.

[31] N. Pérez-Higueras et al. Robot local navigation with learned social cost functions. In *2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 02, pages 618–625, Sept 2014.

[32] Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2096–2101. IEEE, 2016.

[33] George A Quattrone and Edward E Jones. The perception of variability within in-groups and out-groups: Implications for the law of small numbers. *Journal of Personality and Social Psychology*, 38(1):141, 1980.

[34] R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. Transferring human navigation behaviors into a robot local planner. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 774–779, Aug 2014.

[35] Craig Reynolds. Steering Behaviors for Autonomous Characters. In *Game Developers Conference 1999*, 1999.

[36] Rebecca Saxe. Uniquely human social cognition. *Current opinion in neurobiology*, 16(2):235–239, 2006.

[37] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *Symposium on Computer animation*, pages 19–28, 2005.

[38] Emrah Akin Sisbot, Luis F Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.

[39] Henri Tajfel. Social psychology of intergroup relations. *Annual review of psychology*, 33(1):1–39, 1982.

[40] P. Trautman, J. Ma, R.M. Murray, and A. Krause. Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2153–2160, May 2013.

[41] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2153–2160. IEEE, 2013.

[42] J. van den Berg, Ming Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928 –1935, may 2008.

[43] David Wilkie, Jur van den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 5573–5578, Piscataway, NJ, USA, 2009. IEEE Press.

An Egocubemap Based Algorithm for Quadrotors Obstacle Avoidance Using a Single Depth Camera

T. Tezenas Du Montcel*, A. Nègre*, M. Muschinowski*, E. Gomez-Balderas* and N. Marchand*
*Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France

***Abstract*—A fast obstacle avoidance algorithm is a necessary condition to enable safe flights of Unmanned Aerial Vehicles (UAVs) eventually at high-speed. Large UAVs usually have a lot of sensors and available computational resources which allow complex algorithms to run fast enough to navigate safely. On the contrary, small UAVs gather many difficulties, like computation and sensors limitations, forcing algorithms to retain only a few keys points of their environment. This paper proposes an obstacle avoidance algorithm for quadrotor using a single depth camera. Taking advantage of the possibilities offered by embedded GPUs, a cubic world representation centered on the robot - called Egocubemap - is used while the whole obstacle detection and avoidance algorithm is light enough to run at $10\,Hz$ on-board. Numerical and experimental validations are provided.**

I. INTRODUCTION

For some years now, the increase of interest for autonomous navigation of Unmanned Aerial Vehicles (UAVs) has led to the development of many autopilot systems. In order to make a safer one, an avoidance layer is needed under its global navigation control scheme. This is especially true in crowded environments, where the dangerousness of these systems is a big obstacle to concrete applications.

Autonomous obstacle avoidance for UAVs, generally, and quadrotors, especially, is not particularly new [1, 2]. For its simplicity and its low cost, lots of existing works, including ours, focus on a single looking-ahead passive depth sensor [3, 4] which raises three major issues. First, the range of the depth information of those sensors is limited to, typically, distances of 10 meters. Secondly, the angular region of space in which we have depth information is limited to the Field Of View (FOV) of the sensor. Thirdly, whether we use an RGBD camera or whether we apply a stereo-vision algorithm to a pair of stereo-vision images, the extracted depth map is more noisy than a LIDAR point cloud. Those three issues need to be addressed somehow.

The main way to solve the range issue is to have a high frequency obstacle avoidance algorithm to check for obstacles as soon as they enter the perception range. This solution also allows to fly fast with a limited input depth range [3]. We consider that reaching a frequency of $10\,Hz$ is the minimum to allow avoidance at high speed and to give ourselves a chance when considering moving obstacles.

The reduced FOV issue is usually addressed with some kind of onboard memory of the environment. Closely following the idea of Brockers et al. in [5] who presented an Egocylinder type of vision, a cubic world representation centered on the drone - called Egocubemap - is used in this paper to map the environment.

Finally, the lack of precision of the depth map is addressed by implementing a Configuration Space expansion (C-Space) [6] and adding a margin to it. The idea of a C-space expansion is to enlarge any object of the environment by a volume V. At any position in the C-Space expanded environment, it becomes possible to check if there are collisions between the volume V and the environment only by verifying if the center of the volume is occupied by an obstacle. If the chosen volume includes a body, it is therefore possible to check for collisions between this body and the environment in a single operation. Adding a margin in the volume V of the C-Space and making it include the body plus some extra-space will ensure that, if there is no collision in the C-Space, the body is at least at this margin from any obstacle. By adding a margin linked to the error in the depth map, the lack of precision of the depth map will have a reduced impact.

The rest of the paper is organized as follows. In Section II a short review of existing works is given. Section III presents the main contribution of the paper, that is the proposed algorithm. In Section IV, the results obtained during more than $100\,km$ of simulated flight are given. Section V presents the experimental validation in the context of an hardware in the loop implementation. A real UAV is flying and its position obtained by a motion capture system enables us to make it fly in a virtual simulated environment with fictive obstacles. The obstacle avoidance algorithm then runs based on the simulated images. The paper ends with some conclusions and perspectives.

II. RELATED WORKS

Obstacle avoidance consists in a set of functions. If those functions can change depending on the algorithm, two of them are always necessary: creating a world representation from the depth inputs and generating trajectories or direct control commands of the robot. We briefly spoked about other existing avoidance algorithms in the introduction, in this part we will focus on those two important functions of an obstacle avoidance algorithm.

For world representation, there are multiple possibilities. From the simplest one, taking only the raw depth sensor data at the current frame, to the most complex, 3D mapping with SLAM techniques, there is the possibility to keep in memory only a few key points [3, 4], to build a 2D 360° depth representation [5] or to use odometry techniques. There is obviously a tradeoff to find between the computational cost of

a spatial representation and its precision. This is why most of the fastest obstacle avoidance algorithms are keeping only a few key points in memory. In [4], Barry even reduced its stereo algorithm to a single distance pushbroom stereo-vision in order to reduce his processing time. Our goal is to have at least a $360°$ representation of the world while still running above $10\,Hz$ onboard a quadrotor which weights less than $1\,kg$.

For the trajectory generation, there are two main possibilities which are to generate the trajectories on-line or to generate off-line a library of trajectories and to pick one among them on-line. In both cases, trajectories can be linked to a corresponding control to apply on the robot either in open-loop or by implementing some closed-loop control to track it.

In order to generate some trajectories beforehand, it is necessary to discretize the state-space of the quadrotor, and then to generate trajectories for each of the discretized states that allow the quadrotor to go in different directions [7]. Relying on a precise model of their quadrotor, some works even focused on the uncertainty of the generated trajectories [8].

Generating trajectories on-line has one main drawback which is that the generation must be very fast (typically less than a millisecond) in order to be able to generate more than one trajectory and to leave time for the rest of the process. With a precise model, one of the fastest method still uses $2\,ms$ to generate a $500\,ms$ trajectory [9]. It therefore means that it is mandatory to use a simplified model to generate the trajectories. Historically, on-line trajectories have consisted on steering commands or geometric trajectories. Then, Mellinger & Kumar in [10] introduced a method using the differential flatness of the quadrotors which has become the new standard for on-line trajectory generation. Recently, [11] added to a flatness based generation a few efficient tests to check the compatibility of the generated trajectory with a quadrotor dynamic. Due to the constraints of our project which aims at developping a generic, 'plug and play' and working for a wide range of quadrotor, algorithm, we have to generate our trajectories online which is why we adpoted the method from [11].

III. DESCRIPTION OF THE ALGORITHM

The goal is to reach $W_i \in \mathbb{R}^{3\times n},\ i \in [1,n]$ an ordered list of n high level way-points according to the sequence of the ordered list. We consider a way-point W_i reached when the distance between it and $X(t) \in \mathbb{R}^3$, the cartesian position of the quadrotor at time t, is inferior to $\mu_i \in \mathbb{R}^+$, a distance parameter depending on the precision needed by the high level navigation. If no path to a way-point is found, the avoidance algorithm is expected to search for one for 10s before asking the navigation layer for a new high level way-point.

A. Overview of the algorithm

Figure 1 represents the main steps of the proposed algorithm which runs at each new depth acquisition. Its starting point is the depth map acquisition. Many kind of sensors can provide data that are or can be transformed into depth maps. With

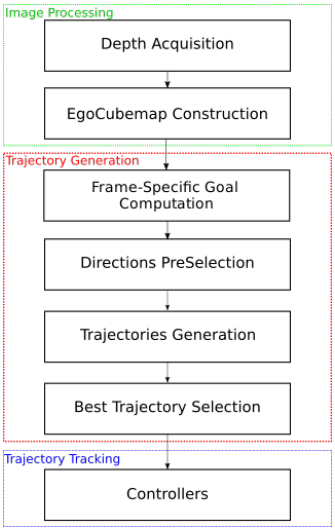


Fig. 1. Overview of the algorithm

stereo-vision cameras, this transformation step consists in running a stereo-vision algorithm, whereas with a direct depth sensor like laser sensors, this step is straightforward. The second step of the algorithm is the construction of the Egocubemap. It starts by the estimation of the displacement between the last two frames. Then an ego centric cubic representation of the environment, the Egocubemap itself, is computed. Finally, it ends with the C-space expansion, which consists in increasing the volume of each depth measure to face unprecise and non dense depth maps. This representation of the environment is the first step to figure out how the next way-point can be reached without colliding with the obstacles. For that purpose, a new frame-specific goal, that may not coincide with the position of the way-point in the frame, is computed. In that frame, trajectories reaching that goal or a neighbourhood of it are generated. Among them, we select the "*best*" one that will be given as input to the closed loop controller of the quadrotor. The following subsections explain those steps in more detail.

B. Egocubemap Construction

The Egocubemap is a world representation shaped in a cube and centered on the quadrotor. Each pixel of the Egocubemap stores the distance from the quadrotor to the closest obstacle in its direction. It is a light dense 360Â° representation of the world. Its construction is as follows:

At each new frame, the ego motion between the last used frame and the new one is estimated. To do so, we use a keyframe-based dense visual odometry mixing the real-time visual odometry from dense RGB-D images detailed in [12] while adding the keyframe feature proposed in [13]. Once the motion is estimated, we move the old Egocubemap according to it. Each pixel from the previous Egocubemap is back projected, displaced from the reverse motion and reprojected to the new estimated Egocubemap. In this step, each pixel is considered as the rectangular area between its corner coordinates in order to have a dense output. There is necessarily some overlapping during the reprojection on the new Egocubemap, which is why

a Z buffer test mechanism is used to keep only the closest depth per pixel.

The new depth data coming from the sensor is then added to the estimated Egocubemap by overwriting the old data with the newly acquired one.

Finally a spheric C-space expansion is applied which means that all the pixels, considered as obstacles, are enlarged by a sphere. We chose to use a spheric C-Space despite the nearly planar volume occupied by a quadrotor considering that it will tilt in space during the flight. Using a sphere allows to check for collisions without the need to recompute the quadrotor angle at each point of the trajectory. To build the C-Space, each pixel is considered as a single point, back projected and enlarged to a sphere. The smallest rectangle overlapping the sphere in the direction of its center is computed (see Figure 2) and reprojected.

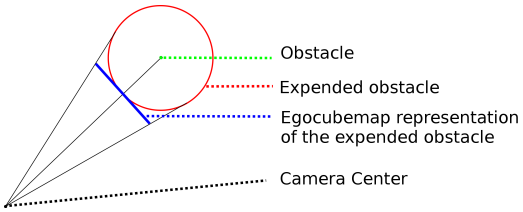
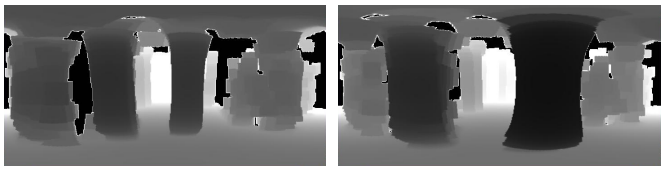


Fig. 2. C-Space reconstruction

The idea behind the C-Space is to check the quadrotor trajectories as its center trajectory instead of the sum of the trajectories of all its components. By reducing the needed checks to only a point instead of a volume at each point of each tested trajectories, we will be able to fasten the checks and therefore to check more trajectories at a reduced cost. 3.a and 3.b are two consecutive planar projections of the Egocubemap enlarged by the C-space.



(a) (b)

Fig. 3. (a) and (b) show two spherical projections of two successive Egocubemaps with Configuration Space

C. Frame-Specific Goal Computation

At this point, we have a representation of our environment, the quadrotor corresponding state and the high level way-point W_i we currently want to reach. We need to define more precisely where we want to go at this specific frame. To do so, we define a new frame-specific goal G. If a trajectory has been defined on the previous frame, the frame-specific goal G is the mean point between the end position of the trajectory and the way-point W_i. By using the previous trajectory in the new goal definition, we stabilize the direction in which we are going and avoid some instabilities in this direction. If no previous trajectory was defined, the frame-specific goal G is simply the high level way-point W_i.

D. Directions Preselection

Now knowing this frame goal G, we select some directions which could potentially lead closer to this goal. To do so, we first truncate the Egocubemap to the distance between our quadrotor and G. We then compute the distance between each point of this truncated Egocubemap and G and finally, among the closest points of the truncated Egocubemap to G, we randomly pick a hundred points E_l. The severity of the restriction on the closeness to G creates a trade off between converging to the frame specific goal and obtaining some diversity in the directions to find at least one path compatible with the flight dynamic.

E. Trajectory Generation

The previously selected points give a hundred different directions that are potentially interesting to reach the goal G and the high level way-point W_i, but we now need to verify how far it is possible to reach in those directions without colliding with the environment. By generating quadrotor feasible trajectories at multiple distances on those directions and projecting them on the Egocubemap, we will be able to check for collisions. To generate the trajectories, we are using the motion primitive generation proposed by Mueller et al. [11]. This method is a trade off between purely geometric methods and dynamically very accurate methods. It creates quadrotors feasible trajectories with the assumption that angular rate commands are tracked perfectly, an assumption which is obviously not exact but from which we are not very far considering the low angular inertia of a quadrotor.

Mueller et al. choose the trajectory as the one that minimizes their cost function, which is the integral of the squared jerk on the trajectory for a given input state, output state and time between those two states. To guarantee the feasibility, it is checked that the required thrust and angular velocity to follow the trajectory are reachable by the quadrotor. Since Mueller et al. found an expression of the minimum of this cost function, this method is really fast and allows the generation of almost a million primitives per second.

It is also worth noting that the cost function can be seen as an upper bound on the average of a product between the thrust and the angular velocity and that it reflects the dynamic difficulty of the trajectory. Therefore, it is interesting to define trajectories that have the same cost because we can expect the precision of the control on those trajectories to be pretty similar. Even if it is not possible to define a cost directly using Mueller et al.'s method, it is still possible to find trajectories with a precise cost using a binary search on the time since the cost depends solely on it for defined input and output states.

In order to avoid obstacles for each of the 100 preselected directions, we try to reach E_l for a few different costs C_m. The lowest cost corresponds to the targeted flight dynamics and the highest cost is chosen at the limit of the quadrotor dynamics in case of emergency. If we can reach E_l at C_m without collision, we generate the next trajectory which is either the same goal E_l with a lower cost or a next goal. If we noticed a collision during the projection of the trajectory on the EgoCubemap, we try to reach a new point E_{l_n} at the same cost C_i. This point is in the direction of the point E_l but at

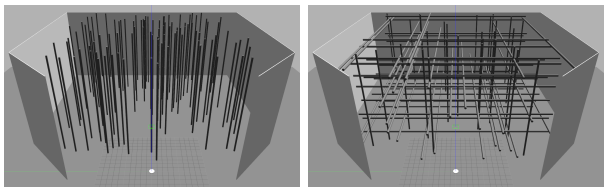


Fig. 4. View of a 2D (100 vertical cylinders) and a 3D (90 vertical or horizontal cylinders) simulated test

a distance reduced by a 0.75 factor. We repeat this operation until $\|E_{l_n} - X(t_k) < min\_dist\|$, with min_dist being the minimum distance of forward progress which depends on the distance between the quadrotor and the frame specific goal.

Finally, all the generated trajectories include a null velocity and acceleration in their final state. This ensures that the last trajectory given to the control will always be a safe one if a hardware failure was to happen. Adding this and the perception limit of 10m creates a velocity limit for the quadrotor for a defined cost. This limit is the one that allows the quadrotor to stop from this initial velocity to a null velocity in 10m for a particular cost. In the following, those velocity limits will be used to characterize the different targeted trajectory cost.

F. Best trajectory Selection

For all obstacle-free generated trajectories, we select the best trajectory $Traj_{Best}$ as a trade off between the trajectory cost $CTraj_l$ and the distance from its final state position $FTraj_l$ to the local goal G_k with more emphasis on the distance :

$$\forall\, l \neq Best, \frac{CTraj_{Best}}{CTraj_l} * \left(\frac{\|G_k - FTraj_{Best}\|}{\|G_k - FTraj_l\|} \right)^2 < 1 \quad (1)$$

In a few cases, we might not find any valid trajectory. This can mainly happen for 2 reasons : an obstacle is closer than min_dist or it is impossible to avoid the impact with a newly detected object (in case of a dynamic object for example). If no valid trajectory exists, we issue a stop command which is treated in a specific way by the controller in order to stop the UAV as fast as possible.

G. Trajectory Tracking

In order to keep our system "plug and play", we use the most common control scheme for quadrotors: a cascade controller. From the trajectory, we use the desired position, velocity and yaw to feed a first PID in position which outputs an acceleration command on the X, Y, Z axes and the desired yaw rate. Using the quadrotor dynamics, the accelerations are converted into the desired thrust, pitch and roll. Those are then fed to a second PID in angle which outputs a pitch rate and a roll rate. The command, which now consists of a thrust, a pitch rate, a roll rate and a yaw rate, can then be mathematically converted into the power needed in each rotor. This control scheme has not been designed to track trajectories and creates errors during the tracking, but it is the most common one. In our project, we wanted to show that even using this control scheme, our algorithm allows to efficiently avoid obstacles. We also keep this control scheme when we issue a stop command but we nullify the proportional term of the PID in position.

H. Computing Time

We implemented all the image processing steps on an embedded GPU card using Cuda and OpenGL. The motion estimation takes $15\,ms$ on an NVIDIA Jetson-TX1, our processing card for onboard computations, while the rest of the process, for a $128\times128\times6$ cubemap, takes $35\,ms$ on the same card. All the trajectory related steps take less than $20\,ms$ on a single CPU core@3.0GHz and less than $55\,ms$ on an NVIDIA Jetson-TX1. The control scheme takes less than $1\,ms$ on both device. Hence in total the algorithm takes less than $70\,ms$ on a ground station and less than $105\,ms$ on an NVIDIA Jetson-TX1. Since we worked only using a ground station, and since we were already above $10\,Hz$ on our ground station, there has been no emphasis on improving the performances of the algorithm, especially for all the trajectory related steps. By parallelizing the trajectory related steps, whether on the GPU or on the CPU, we feel confident about reaching $10\,Hz$ performances on an NVIDIA Jetson-TX1, thus enabling real-time aggressive motion planning.

IV. SIMULATION

A. The Setup

We decided to work with ROS since it is very widely used and allows easily to exchange packages in the same "plug and play" spirit that we followed. The use of the Gazebo simulator was then pretty straight forward considering that it had all the features we needed and the quality of its ROS integration.

We are using a simulator based on the fcu_sim ROS package from BYU-MAGICC. This package offers Gazebo/ROS sensors plug-ins and a quadrotor dynamics simulator. The idea of this simulator is to define the quadrotor as a simple 6DOF rigid body on Gazebo and then to add the forces, torques and saturations that differ between a quadrotor and a 6DOF rigid body of same mass and inertia as described in [14].

Our simulated quadrotor has a radius of $0.5\,m$, weights $3.52\,kg$. This quadrotor is around five times heavier and two times larger than the one we will use when doing real hardware experimentations, but our algorithm is supposed to be scalable and we did not want to change the native model which would have probably led to more approximation on the modeling.

The tests have been designed so that the previously defined quadrotor travels 100m in an unknown environment filled with randomly positioned obstacles. The obstacles consist in 16cm radius cylinders which cross the whole scene in specific axes. The tests are run 100 times with different maximum velocities and obstacle number which means that the quadrotor flies at least $10\,km$ in each configuration. Taking into account the fact that, in the earth frame, the quadrotor dynamics on the Z axis are different from the dynamics on the X and Y axes, which are similar due to symmetries, we designed two different tests. The first one involves only the X and Y axes (2D test) while the other one involves all three axes (3D test). The only difference between both tests is the direction of those cylinders which is only along the Z-axis in the 2D case and which is along the X, Y or Z axes in the 3D test. On Figure 4 are represented specific configurations of a 2D and a 3D test with respectively 100 and 90 cylinders.

Test directions	2D	2D	2D	3D	3D	3D
Obstacle Number	50	100	150	30	90	150
Max Velocity: 3,3 m/s						
Collisions/km	0.0	0.1	0.8	0.0	0.1	0.3
Max Velocity: 5 m/s						
Collisions/km	0.1	0.4	0.7	0.0	0.1	1.0

TABLE I
RESULTS OF THE TESTS IN SIMULATION USING THE SIMPLE MODEL.

B. Results

Figure 5 is a view from above of the quadrotor trajectory from a typical case of a test with 100 vertical (2D test) cylinders.

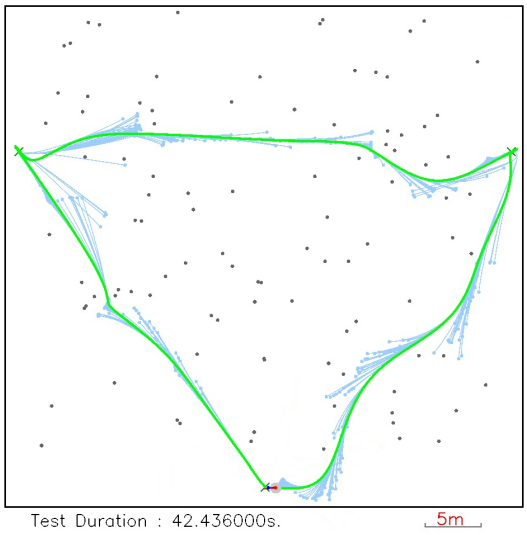


Fig. 5. A view from above of the quadrotor trajectory during a simulated 2D test with 100 cylinders. In green, the quadrotor trajectory. In blue, the trajectories generated each time the avoidance algorithm is called.

Table I gives the results of the different configurations tested in our simulated environment. The different maximum velocities have been defined depending on the time horizon of the perception of the quadrotor. With a $3.33\,ms^{-1}$ maximum velocity, given a $10\,m$ max depth perception, the time horizon is at least $3\,s$. That duration is reduced to $2\,s$ with a $5\,ms^{-1}$maximum velocity. The C-Space radius was 70cm which constitutes a margin corresponding to 40% of the quadrotor radius.

There are two main results :

- Reaching very low or null errors in multiple cases, even in very crowded environments, validates our choices while building the algorithm.
- In the very crowded 2D 150 obstacles environments, our algorithm get stuck in local minimums. In all cases, the quadrotor stops in front of the obstacles during 10s, as expected, after which it could be tasked to land.

V. EXPERIMENTAL VALIDATION

A. The Setup

The experimental validation was carried out on a homemade quadrotor represented on Fig. 6. It weights 303g and has a 33cm diameter including its 5" blades. It embarks brushless motors, a NAZE32 flight controller flashed with ROSflight [15] and it is powered by a 7.4V LIPO battery. The NAZE32 IMU was used to feed the ROSflight attitude controller and we used a motion capture system (MOCAP) to feed the avoidance algorithm and the position controller on the quadrotor state.

Fig. 6. A photo of our custom quadrotor

For the tests, all the avoidance related computations are done on a ground station. Embedding a NVIDIA Jetson-like processing card for onboard computations would necessitate a bigger frame, which is impossible in our motion capture room which useful volume is a 3m×2.5m×2m cuboid. The quadrotor is not equipped with any depth sensor. To provide such measurements to the quadrotor, a virtual clone of the quadrotor, with the same position and orientation as the real one in the MOCAP room, evolves in a Gazebo world. A depth image can then be created in this virtual environment. Virtual obstacles or clones of the real ones can be added to this world. Due to the limited size of the MOCAP volume compared to the quadrotor size, we could only create a scene with 6 or less cylinders. As in simulation, the cylinders are randomly spawned.

B. Results

Figure 7 is a summary from above of an hardware-in-the-loop test with 6 cylinders.

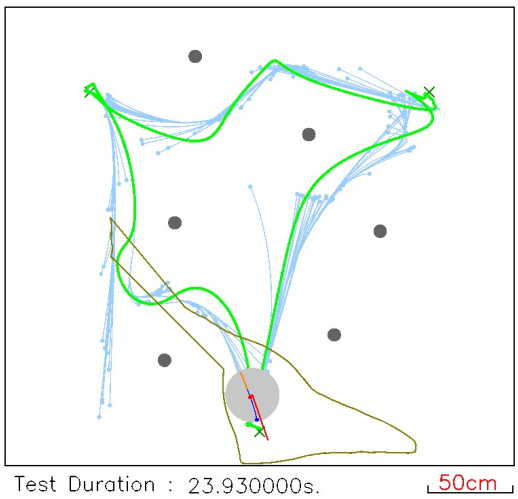


Fig. 7. A view from above of the quadrotor trajectory from a hardware-in-the-loop test with 6 vertical cylindrical obstacles. In green, the quadrotor trajectory. In blue, the trajectories generated each time the avoidance algorithm is called

Table II gives the results of the hardware in the loop tests.

Test directions	2D	2D	2D
Max Theorical Velocity	3.3 m/s	5 m/s	3.3 m/s
Number of obstacles	4	4	6
Number of tests	10	10	10
Successful tests	10	10	10
of which local minimum stops	1	2	3

TABLE II
RESULTS OF THE HARDWARE IN THE LOOP TESTS

Due to the size limitation of the MOCAP room, we were only able to test with a maximum of 6 vertical obstacles. Even with this few obstacles, 20% of the tests ended in a local minimum but it's worth noticing that the quadrotor correctly hovered, waiting for a new high-level waypoint. Also, since the algorithm has a zero velocity and acceleration constraint at the end of each generated trajectory, because of the size limitation of the MOCAP room and the short trajectories, the maximum velocity of the quadrotor was never above 1m/s despite theoretically being able to go above it in the tested configurations. Testing it in larger environment will increase the velocity of the quadrotor and is clearly the next step we have to take. Despite this limitation, the algorithm reacted as expected during those flights and there have been only a few differences between the simulated results and the hardware-in-the-loop ones. The main difference resides in the quality of the trajectory tracking. Even at those low velocity, the cascade PID scheme shows some of its limit due to an 80ms latency in the loop. We expect this delay to be reduced in a fully embedded scenario and therefore an improvement of the control.

VI. CONCLUSIONS

We presented an obstacle avoidance algorithm solely based on a single facing ahead depth input with a field of view corresponding to what is expected from a pair of stereo vision cameras. Using an Egocubemap world representation, we successfully and consistently avoided obstacles whether in simulation or in a hardware-in-the-loop setup and in differently crowded environments. Our next step will be to attempt outdoor and fully embedded flights with our algorithm.

Due to the uniqueness of the tests of each published paper, comparing our results to other existing works is also really complicated at the moment. In order to make this easier, we are currently creating an avoidance benchmark. We believe that it could help highlighting on the strengths and weaknesses of each approach.

VII. ACKNOWLEDGMENT

This work is founded by the CAP2018 project.

REFERENCES

[1] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying fast and low among obstacles: Methodology and experiments," *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 549–574, 2008. [Online]. Available: https://doi.org/10.1177/0278364908090949

[2] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 807–814.

[3] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 50–56.

[4] A. J. Barry, "High-speed autonomous obstacle avoidance with pushbroom stereo," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[5] R. Brockers, A. Fragoso, B. Rothrock, C. Lee, and L. Matthies, "Vision-based obstacle avoidance for micro air vehicles using an egocylindrical depth map," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 505–514.

[6] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE transactions on computers*, no. 2, pp. 108–120, 1983.

[7] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, "Robust monocular flight in cluttered outdoor environments," *arXiv preprint arXiv:1604.04779*, 2016. [Online]. Available: https://arxiv.org/pdf/1604.04779.pdf

[8] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017. [Online]. Available: https://doi.org/10.1177/0278364917712421

[9] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1398–1404.

[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[11] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec 2015.

[12] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, pp. 719–722.

[13] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2100–2106.

[14] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, Feb 2014.

[15] J. Jackson and D. Koch, "ROSflight," 2016. [Online]. Available: http://rosflight.org/

DisNet: A novel method for distance estimation from monocular camera*

Muhammad Abdul Haseeb, Jianyu Guan, Danijela Ristić-Durrant, Axel Gräser, *Member, IEEE*
Institute of Automation, University of Bremen, Otto-Hahn-Allee NW1, 28359 Bremen, Germany

***Abstract*— In this paper, a machine learning setup that provides the obstacle detection system with a method to estimate the distance from the monocular camera to the object viewed with the camera is presented. In particular, the preliminary results of an on-going research to allow the on-board multisensory system, which is under development within H2020 Shift2Rail project SMART, to autonomously learn distances to objects, possible obstacles on the rail tracks ahead of the locomotive are given. The presented distance estimation system is based on Multi Hidden-Layer Neural Network, named DisNet, which is used to learn and predict the distance between the object and the camera sensor. The DisNet was trained using a supervised learning technique where the input features were manually calculated parameters of the object bounding boxes resulted from the YOLO object classifier and outputs were the accurate 3D laser scanner measurements of the distances to objects in the recorded scene. The presented DisNet-based distance estimation system was evaluated on the images of railway scenes as well as on the images of a road scene. Shown results demonstrate a general nature of the proposed DisNet system that enables its use for the estimation of distances to objects imaged with different types of monocular cameras.**

I. Introduction

Reliable and accurate detection of obstacles is one of the core problems that need to be solved to enable autonomous driving. In the past decades, significant work has been done to address the problem of obstacle detection [1][2]. Besides the emerging of novel algorithms, technology development also enables progress in autonomous obstacle detection. Different onboard sensors such as radars, mono/stereo cameras, LIght Detection And Ranging - LiDAR, ultrasonic sensors and others, implemented in so-called Advanced Driving Assistance Systems (ADAS), are rapidly increasing the vehicle's automation level [3][4].

Many approaches have been presented for different application fields and scenarios. Whereas other transport modes have been quick to automate certain operations, rail runs the risk of lagging behind. One of the key challenges, which has so far hindered automation of rail systems, is the lack of a safe and reliable onboard obstacle detection system for trains within existing infrastructure [5]. In recent years, there is a tendency to use experience from obstacle detection both in the automotive and the aviation sector for the development of autonomous obstacle detection in railways [6]. While the main principle of obstacle detection in front of a vehicle from the automotive sector can be applied to railway applications, there are also specific challenges. One of the key challenges is long-range obstacle detection. Sensor technology in current land transport research is able to look some 200 m ahead [7]. The required rail obstacle detection interfacing with loco control should be able to look ahead up to 1000 m detecting objects on and near track which may potentially interfere with the clearance and ground profile.

The method for long-range obstacle detection presented in this paper is developed within project "SMART-SMart Automation of Rail Transport", funded by the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme [8]. The main goal of this project is to increase the effectiveness and capacity of rail freight through the contribution to automation of railway cargo haul at European railways by developing of a prototype of an autonomous Obstacle Detection System (ODS). Project SMART will contribute to the long-term vision for an autonomous rail freight system, by the development, implementation and evaluation of a prototype integrated on-board multi-sensor system for reliable autonomous detection of potential obstacles on rail tracks, which could assist drivers and in long term could be used for autonomous initialization of braking of the freight train.

As illustrated in Fig. 1, the SMART ODS combines different vision technologies: thermal camera, night vision sensor (camera augmented with image intensifier), multi stereo-vision system (cameras C1, C2 and C3) and laser scanner (LiDAR) in order to create a sensor fusion system for mid (up to 200 m) and long range (up to 1000 m) obstacle detection, which is independent of light and weather conditions.

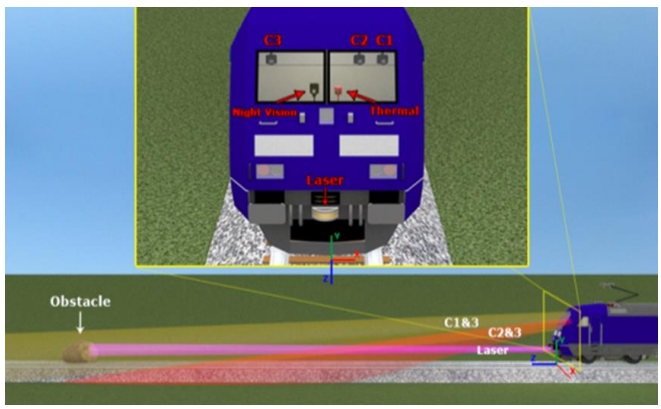


Figure 1. Concept of the SMART multi-sensor ODS. (Top) Front view of the sensors mounted on a locomotive. (Bottom) Side view of the range sensors and an obstacle detection scene.

*Research supported by Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 730836.

M. A. Haseeb, J. Guan, D. Ristić-Durrant and A. Gräser are with the Institute of Automation, University of Bremen, Otto-Hahn-Allee, NW1, Bremen 28359, Germany (Corresponding author. Tel.:+49-421-218-62446; fax: +49-421-218-9862446; e-mail: haseeb@iat.uni-bremen.de).

The main idea behind the multi-sensory system is to fuse the sensor data as sensors individually are not yet powerful enough to deal with complex obstacle detection tasks in all the SMART defined application scenarios, which include day and night operation and operation in poor visibility condition. Because of this, the development of an adequate data fusion system, which effectively combines data streams from multiple sensors, is required. The data fusion approach will be designed based on sensor data availability. Namely, independently of the illumination condition, sensor data from the thermal camera and laser scanner will be always available, where the implemented laser scanner data will be reliably available only in the certain range of up to 100 m. In contrast to that, the stereo camera system fails to generate data under poor illumination conditions, and the night vision camera cannot operate during the day. After obtaining fused data, based on the individual advantages of each sensor, the resulting data stream will be used for detection of obstacles on the rail tracks and for calculation of the distances from the locomotive to detected obstacles. While for stereo cameras traditional depth extraction can be used for thermal camera and night vision camera, estimation of distances from single camera shall be performed.

In this paper, initial results on object distance estimation from monocular cameras are shown using a novel machine learning based method named as DisNet – a multilayer neural network for distance estimation. Although the presented method has been originally developed for autonomous obstacle detection in railway applications, it can be applied to road scenes as well, as it is illustrated in the evaluation section of this paper.

II. Related Work

One of the crucial tasks in autonomous obstacle detection nowadays is finding the solutions to the combination of the environment perception sensors, where vision-based obstacle detection is still considered irreplaceable [9]. Besides its cheaper price, vision is also known as much evolving technology where most of its data is usable as compared to radar and LiDAR [10].

Obstacle detection in computer vision is most commonly done via stereo vision, in which images from two stereo cameras are used to triangulate and estimate distances to objects, potential obstacles, viewed by cameras [11]. Besides the individual use of stereo vision, in a number of obstacle detection systems stereo vision is combined with other range sensors. For example, in [3], an obstacle detection system was developed based on a fusion system consisting of computer vision and laser scanner. The laser provided a point cloud (PC) from which the system extracted the obstacles (clusters of points). These clusters were used both for the region of Interest (ROI) generation for computer vision and as information for obstacle classification, based on machine learning.

Beyond stereo/triangulation cues, there are also numerous monocular cues such as texture variations and gradients, defocus, and colour/haze, which contain useful and important depth information. Some of these cues apply even in regions without texture, where stereo would work poorly. Because of this, some authors follow the idea of human perception of depth by seamlessly combining many of stereo and monocular cues. In [1], a Markov Random Field (MRF) learning algorithm to capture some of these monocular cues is applied, and cues are incorporated into a stereo system. It was shown that by adding monocular cues to stereo (triangulation) ones, significantly more accurate depth estimates than is possible using either monocular or stereo cues alone is obtained. In [13], supervised learning to the problem of estimating depth maps only from a single still image of a variety of unstructured environments, both indoor and outdoor, was applied. However, depth estimation from a single still image is a difficult task, since depth typically remains ambiguous given only local image features. Thus, the presented algorithm must take into account the global structure of the image, as well as use prior knowledge about the scene.

In this paper, a novel method for object distance estimation from a single image, which does not require either a prior knowledge about the scene or explicit knowledge of the camera parameters, is presented. The presented distance estimation system is based on Multi Hidden-Layer Neural Network, named DisNet, which is used to learn and predict the distance between the object and the camera sensor.

III. Neural Network-Based Object Distance Estimation From Monocular Camera

The architecture of the DisNet-based distance estimation system is illustrated in Fig. 2. The camera image is input to the Object Classifier which is based on a state-of-the-art computer vision object detector YOLO (You Only Look Once) [14] trained with COCO dataset [15]. YOLO is a fast and accurate object detector based on Convolution Neural Network (CNN). Its outputs are bounding boxes of detected objects in the image and labels of the classes detected objects belong to. The objects bounding boxes resulted from the YOLO object classification are then processed to calculate the features, bounding boxes parameters. Based on the input features, the trained DisNet gives as outputs the estimated distance of the object to the camera sensor. In the system architecture illustrated in Fig. 2, an example of the estimation of distances of two persons on the rail tracks is shown.

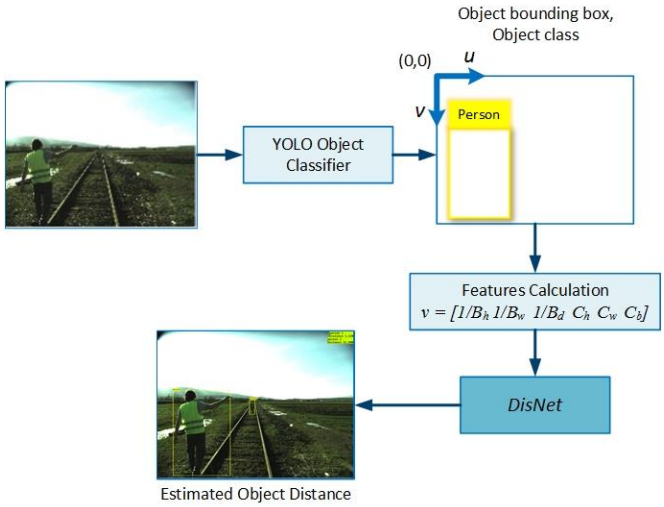


Figure 2. The DisNet -based system used for object distance estimation from a monocular camera

For the training of DisNet, a supervised learning technique was used. This method required a collected dataset including both inputs and outputs, i.e. the ground truth. In the presented system, training dataset was collected manually by manual extraction of 2000 bounding boxes of different objects in the images recorded by RGB cameras at different distances together with the ground truth, which was the accurate laser scanner measurement of the distances to objects in the recorded scene. The details of the structure and training of DisNet are given in the following sections.

A. *DisNet training - Dataset*

In the presented work, the objective is that DisNet is trained for the estimation of an object's distance to the onboard sensory obstacle detection system. More formally, the task is to estimate the distance to an object in the laser's reference frame, which is on the same distance from the object as the camera reference frame, given an input also called feature vector $\boldsymbol{v}$. In the presented work, $\boldsymbol{v}$ contains the features of the bounding box of the object detected in camera images and the ground-truth is the distance to the object as measured by the laser scanner.

In order to build the dataset, the objects positions and their bounding boxes in the RGB images were manually extracted and 2000 input feature vectors were created. In order to achieve sufficient discriminatory information in the dataset, different objects at different distances, which could be present in a railway scene as possible obstacles on the rail tracks, were considered. Some of the objects recorded at different distances and their bounding boxes from the dataset are shown in Fig. 3.

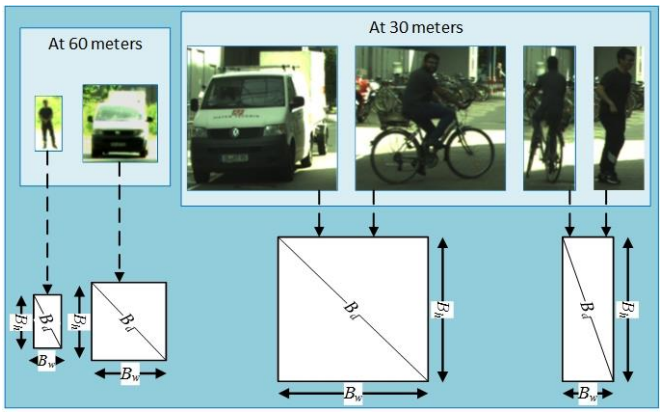


Figure 3. Examples from the DisNet dataset of different object bounding boxes in the RGB images

For each extracted object bounding box, a six-dimensional feature vector $\boldsymbol{v}$ was calculated:

$$\boldsymbol{v} = [1/B_h \; 1/B_w \; 1/B_d \; C_h \; C_w \; C_b] \quad (1)$$

where the coordinates of vector $\boldsymbol{v}$, features, are:

Height, B_h=(height of the object bounding box in pixels/image height in pixels)
Width, B_w=(width of the object bounding box in pixels/image width in pixels)
Diagonal, B_d=(diagonal of the object bounding box in pixels/image diagonal in pixels)

The ratios of the object bounding box dimensions to the image dimensions B_h, B_w and B_d enable the reusability of DisNet trained model with a variety of cameras independent of image resolution. C_h, C_w and C_b in (1) are the values of average height, width and breadth of an object of the particular class. For example for the class "person" C_h, C_w and C_b are respectively 175 cm, 55 cm and 30 cm, and for the class "car" 160 cm, 180 cm and 400. The features C_h, C_w and C_b are assigned to objects labelled by YOLO classifier as belonging to the particular class in order to complement 2D information on object bounding boxes and so to give more information to distinguish different objects.

The relationships of the calculated features of object bounding boxes in 2D image, B_h, B_w and B_d, and the real distance to the image measured by laser scanner in the range 0-60 m, are given in Fig. 4. Geometrically, by the projective transformations, the object bounding box size is expected to get smaller the further away the object is, so the inverse of bounding box size is expected to increase as the distance increases. Inspection of the data confirms that this is the case and suggests that the relationship is approximately linear, which gives a clear motive to use it for the dataset used for training of DisNet.

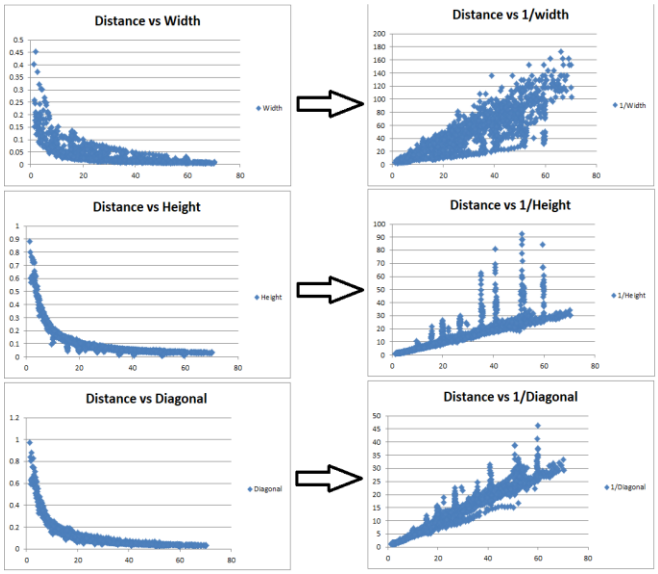


Figure 4. DisNet features vs. distance

For training the network the input dataset was firstly randomly split into a training (80% of the data), validation (10% of the data) and test set (10% of the data).

The DisNet was trained using the backpropagation method with Adam optimizer [16] on the dataset collected.

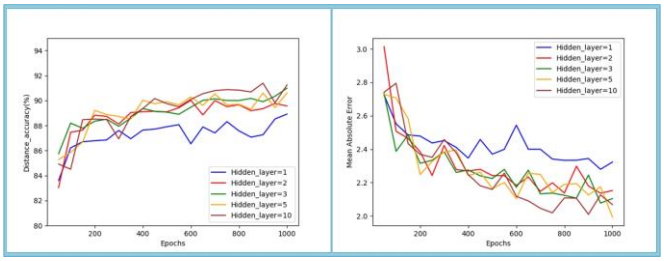
B. *DisNet structure*

In order to find the appropriate number of hidden layers experiments with various numbers of hidden layers (1,2,3,5 and 10) were performed assuming that each hidden layer had 100 neurons. Fig. 5 (a) shows the accuracy of distance estimation over 1000 epochs achieved for different number of hidden layers. As obvious, DisNet with one hidden layer achieves the lowest distance estimation accuracy. It is also obvious that there is no significant difference in distance

estimation accuracy achieved with 2,3,5 and 10 hidden layers. For this analysis, a reduced dataset was used. The networks were trained on the 80% dataset and the estimation accuracy reported is on the 10% validation set.

Similar behaviour can also be seen in Fig. 5(b) where the Mean Absolute Error over 1000 epochs achieved for a different number of hidden layers is shown. As obvious, the Mean Absolute Error is largest for the DisNet with one hidden layer, while there is no significant difference in the Error achieved with 2,3,5 and 10 hidden layers.

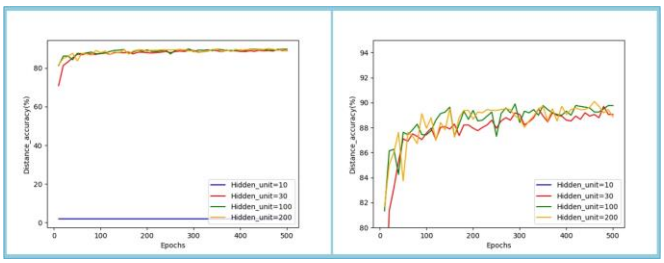
Even though the smallest values of Mean Absolute Error were achieved for 10 hidden layers and the distance accuracy was highest for 10 hidden layers, a trade-off was made between the computational time and accuracy/error and finally, DisNet with 3 hidden layers was chosen.



(a) (b)

Figure 5. (a) Distance Estimation Accuracy and (b) Mean Absolute Error achieved for different numbers of hidden layers

After making a decision on network with 3 hidden layers, in order to find the appropriate number of neurons for the hidden layers experiments with various numbers of hidden neurons were performed. Fig. 6 (a) shows the accuracy of distance estimation over 1000 epochs achieved for different number of neurons per hidden layer. As obvious, the distance estimation accuracy achieved with 10 hidden neurons is very low, much lower than distance estimation accuracy achieved with 30, 100 and 200 hidden neurons. The magnified diagram in Fig. 6 (b) shows that distance estimation accuracy with 30 hidden neurons is lower than with 100 and 200 neurons. Bearing in mind that there is no significant difference in distance accuracy estimation with 100 and 200 hidden neurons, in order to reduce the complexity of DisNet, finally, 100 neurons per hidden layer were chosen.



(a) (b)

Figure 6. Distance Estimation Accuracy achieved for different number of hidden neurons per hidden layer in 3-hidden layers neural network DisNet

The final structure of DisNet having 3 hidden layers with 100 hidden neurons per layer is shown in Fig. 7.

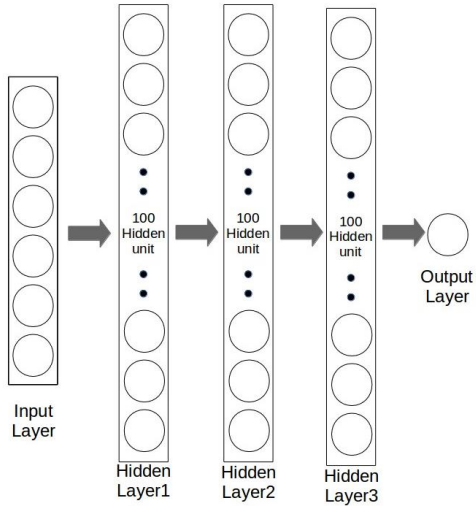


Figure 7. The structure of DisNet used for object distance prediction

DisNet input layer consists of 6 neurons corresponding to 6 features, parameters of output layer consists of only one single neuron. The output of this node is the estimated distance between the camera and the object viewed with the camera.

IV. Evaluation

The DisNet-based system for distance estimation was evaluated on images recorded in the field tests within the H2020 Shift2Rail project SMART [8]. The sensor data, which were used for the evaluation of a DisNet-based system for object distance estimation, were recorded in the field tests on the straight rail tracks in different times of the day and night on the location of the straight rail tracks (Fig. 8). Monocular RGB cameras were mounted on the static test-stand, together with the laser scanner in the locations which resemble their intended locations in the final integrated SMART obstacle detection (Fig. 8). During the performed field tests, the members of the SMART Consortium imitated potential static obstacles on the rail tracks located on different distances from the SMART test-stand.

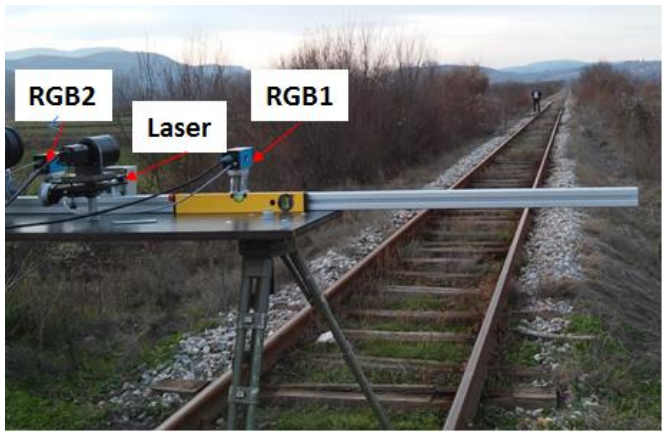


Figure 8. Field tests performed on the straight rail tracks; Test-stand with the SMART sensors viewing the rail tracks and an object (person) on the rail track

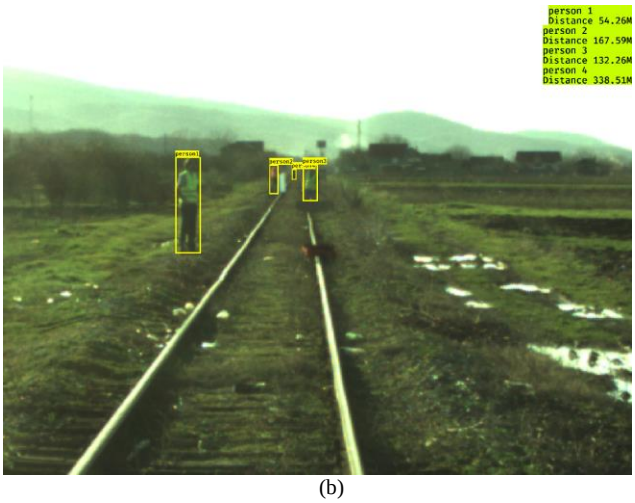
A. *Railway Scene - Distance estimation from the single RGB camera image*

Some of the results of the DisNet object distance estimation in RGB images are given in Fig. 9. The estimated distances to the objects (persons) detected in the images are given in Table I.

TABLE I. ESTIMATED DISTANCES VS. GROUND TRUTH

Figure	**Object**	**Rail Scene**	
		Ground Truth	***Distance estimated from DisNet***
9 (a)	Person 1	100 m	101.89 m
	Person 2		99.44 m
9 (b)	Person 1	50 m	54.26 m
	Person 2	150 m	167.59 m
	Person 3	100 m	132.26 m
	Person 4	300 m	338.51 m

(a)



(b)

Figure 9. DisNet estimation of distances to objects in a rail track scene from the RGB camera image. (a) Distance estimation of detected persons at 100 m and (b) Magnified RGB image overlaid with bounding boxes and distance estimation of detected persons at 50, 100, 150 and 300 m respectively.

As obvious from Fig. 9, YOLO based object detection in images is reliable in spite of the fact that YOLO classifier was used in its original form trained with COCO dataset [15], without re-training with the images from the SMART field tests. Also, it is obvious that achieved distance estimation is satisfactory in spite of the fact that DisNet database did not contain object boxes from the real rail tracks scenes. This, in the first place, means that the objects in real field tests scenes were at larger distances from the sensors than in the recording tests used for dataset building. Also, the distances of the objects in field tests were outside the laser scanner range used for the training of DisNet. The difference in estimation of distances of persons at 100 m (Fig. 9(a) and 9(b)) indicates the need for improvement of objects bounding boxes extraction. Namely, the person at 100 m in Fig. 9(b) is not fully bounded with the bounding box as the lower part of the person is occluded by a board. Also, in future work, novel features with a higher correlation score with respect to distance will be investigated and will be used to improve the accuracy of distance estimation in SMART obstacle detection system.

A. *Road Scene - Distance estimation from the single RGB camera image*

Although presented DisNet-based method for distance estimation from the monocular camera has been originally developed for autonomous obstacle detection in railway applications, it can be applied to road scenes as well. To demonstrate this, presented method was applied to the image of a different resolution than images used for training of DisNet. The image of a road scene was recorded within the project HiSpe3D-Vision presented in [11][17]. The main goal of HiSpe3D-Vision was to develop a high speed, low latency stereo vision based collision warning system for automotive applications. The obstacle detection and distance calculation for collision warning were based on the segmentation of the disparity map created from the car-mounted stereo-vision system. The result of object detection and distance estimation in a dynamic environment (moving car and moving object-obstacle) is shown in Fig. 10, where original image is overlaid with the bounding cuboid for the object closest to the car (person on the bike). Distance for this object, as estimated by the HiSpe3D-Vision method, is given in the left upper corner of the image in Fig. 10, as well as in Table II.

In contrast to HiSpe3D-Vision method, which detected only the object closest to the car, the presented DisNet method recognized different objects in the scene recorded by the car-mounted camera: person, bicycle, car and track. The bounding boxes of the recognized objects are overlaid on the image in Fig. 10 together with distances estimated by DisNet. The objects distance estimation achieved by DisNet vs. the distance estimation achieved by HiSpe3D stereo vision method is given in Table II.

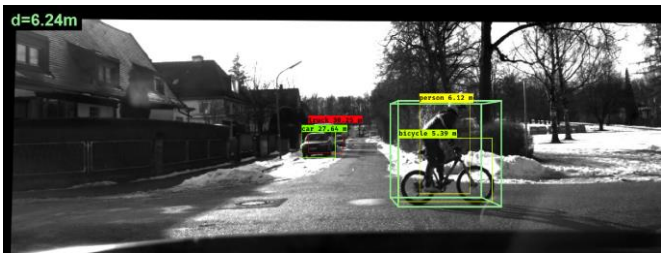


Figure 10. Road scene image overlaid with objects recognition and distance estimation results achieved by proposed DisNet and by stereo-vision based HiSpe3D method [17]

TABLE II. OBJECTS DISTANCES ESTIMATED BY DISNET VS. OBJECTS DISTANCES ESTIMATED BY HISPE3D-VISION METHOD [17]

Object	**Road Scene**	
	Distance estimated by HiSpe3D-Vision	***Distance estimated by DisNet***
Person	6.24 m	6.12 m
Bicycle	-	5.39 m
Car	-	27.64 m
Truck	-	30.25 m

As obvious, DisNet outperforms HiSpe3D-Vision method in a number of different objects recognized in the recorded scene. The person distance estimation by both methods is comparable. The difference in distances for the person and the bicycle, estimated by DisNet, indicates the need for improvement of objects bounding boxes extraction. In future work, the YOLO classifier will be replaced with 2D image processing based classifier and bounding box extractor, which is under development in the SMART project.

V. CONCLUSION

In this paper, the initial results of DisNet – a machine learning-based distance estimation from the monocular camera, achieved by obstacle detection system under development within Shift2Rail project SMART-Smart Automation of Rail Transport, are presented. Presented results illustrate reliable estimation of distances from a single RGB camera to objects in static railway scenes recorded by cameras. General nature of the presented distance estimation method is demonstrated by the result of distance estimation in a dynamic road scene captured with different types of cameras. This indicates that in future work presented method can be used for object distance estimation from different types of monocular cameras integrated into the SMART on-board obstacle detection system, thermal camera and night vision camera. Further, the presented obstacle detection system will be evaluated in dynamic field tests when mounted on a locomotive in motion. The presented results from dynamic road scene justify the expectation that DisNet-based obstacle detection system will work on real experimental images when the train is in motion.

ACKNOWLEDGEMENT

This research has received funding from the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 730836.

REFERENCES

[1] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, M. Sabbatelli, Real-Time Obstacle Detection using Stereo Vision for Autonomous Ground Vehicles: A Survey, 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), China.

[2] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in IEEE Int. Conf. on Computer Vision and Pattern Recognition, 2012

[3] F. Jiménez, J. E. Naranjo, J. J. Anaya, F. García, A. Ponz, J. M. Armingol, Advanced Driver Assistance System for road environments to improve safety and efficiency, Transportation Research Procedia 14 (2016) 2245 – 2254.

[4] S. Kim, H. Kim, W. Yoo, K. Huh, Sensor Sensor Fusion Algorithm Design in Detecting Vehicles Using Laser Scanner and Stereo Vision, IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 4, 2016.

[5] Shift2Rail Joint Undertaking, Multi-Annual Action Plan, Brussels, November 2015.

[6] J. Weichselbaum, C. Zinner, O. Gebauer, W. Pree, Accurate 3D-vision-based obstacle detection for an autonomous train, Computers in Industry 64 (2013), pp. 1209–1220.

[7] P. Pinggera, U. Franke, R. Mester, High-performance long range obstacle detection using stereo vision, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2015).

[8] Shift2Rail project SMART: http://www.smartrail-automation-project.net

[9] F. de Ponte Müller, Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles, Sensors 2017, 17(2).

[10] J. Park, J.-H. Lee, S. H. Son, A Survey of Obstacle Detection using Vision Sensor for Autonomous Vehicles, 2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications.

[11] A. Leu, D. Aiteanu, A. Gräser, High Speed Stereo Vision Based Automotive Collision Warning System, Applied Computational Intelligence in Engineering and Information Technology, Volume 1, pp. 187-199, Springer Verlag Heidelberg, 2012.

[12] A. Saxena., J. Schulte, A. Y. Ng, Depth estimation using monocular and stereo cues. In: IJCAI, 2007.

[13] A. Saxena, H.·Sung, A. Y. Ng, 3-D Depth Reconstruction from a Single Still Image; Int J Comput Vis, 2007.

[14] Redmon, Joseph and Farhadi, Ali, *YOLOv3: An Incremental Improvement*, arXiv, 2018.

[15] COCO dataset, https://arxiv.org/pdf/1405.0312.pdf

[16] D. P. Kingma, J. L.Ba, ADAM: A Method for Stochastic Optimization,https://arxiv.org/pdf/1412.6980.pdf

[17] A. Leu, D. Bacără, D. Aiteanu, A. Gräser, Hardware acceleration of image processing algorithms for collision warning in automotive applications, Methods and Applications in Automation: 32nd – 33rd Colloquium of Automation, Salzhausen/Leer, Germany, pp.1-12, Shaker Verlag GmbH, 2012.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session V

Control, Planning

- **Keynote speaker: Jose Eugenio Naranjo Hernandez (Universidad Politecnica de Madrid, Spain)**
  Title: Integration of Cooperative Services with Autonomous Driving

Title: Multi-Sensor-Based Predictive Control for Autonomous Backward Perpendicular and Diagonal Parking
Authors: D. Perez-Morales, O. Kermorgant , S. Domınguez-Quijada and P. Martinet

Title: Towards Uncertainty-Aware Path Planning for Navigation on Road Networks Using Augmented MDPs
Authors: L. Nardi, C. Stachniss

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session V

Keynote speaker: **Jose Eugenio Naranjo Hernandez**
(Universidad Politecnica de Madrid, Spain)

Title: Integration of Cooperative Services with Autonomous Driving

Abstract: Cooperative Systems (C-ITS) are based on the generation of safety and efficiency information in road transport and its diffusion through the use of V2X vehicular communications networks. Several C-ITS experiences have been developed, which were mainly focused on sending traffic information via Vehicle-to-Infrastructure (V2I) communications to vehicles, in such a way that the same information shown in the variable information panels is presented to the driver inside the vehicle through a Human Machine Interface (HMI). Also, there are experiences where there is an exchange of information between some vehicles and others using vehicle-to-vehicle communications services (V2V). In this way, the limit of the visual horizon of the vehicles is exceeded. drivers and information on the entire driving environment that may affect safety.

The European ITS Platform has published the first two sets of cooperative services that are currently available for deployment and have been named Day 1 and Day 1.5, in reference to their implementation term. In this way, within the C-ITS Day 1 services we can find the Emergency vehicle approaching (V2V), Hazardous location notification (V2I), the Road works warning (V2I) or the Weather conditions (V2I), and within the services C-ITS Day 1.5 we found Cooperative collision risk warning (V2V) or Zone access control for urban areas (V2I). This set of services is currently being deployed in projects such as C-ROADS (https://www.c-roads.eu), where C-ITS corridors are being enabled across Europe.

On the other hand, it is clear that autonomous driving in the strict sense, at an SAE-3 or higher automation level, does not make sense if cooperative and connectivity components are not incorporated within the ego-vehicle, since the limitation of the horizon visual sensors on the vehicles makes them subject to the limitations of human beings. In this keynote, we present the results of the European project AUTOCITS – Regulation Study for Interoperability in the Adoption of Autonomous Driving in European Urban Nodes (https://www.autocits.eu/), where they try to obtain synergies of both C-ITS as of the autonomous and connected vehicles, enabling the Cooperative Autonomous Driving. In this way, AUTOCITS is developing all the architecture to allow the TMCs to generate C-ITS Day 1 messages and transmit them to the road via V2X communications, which have been deployed in three pilot sites located in the urban accesses of 3 European cities: Paris, Madrid and Lisbon, belonging to the Atlantic Corridor of the TEN-T road network.

Biography: José E. Naranjo was born in Orense, Spain, in 1975. He received the B.E., M.E., and Ph.D. degrees in computer science engineering from the Technical University of Madrid (UPM), Madrid, Spain, in 1998, 2001, and 2005, respectively. From 1999 to 2007, he was with the Instituto de Automática Industrial, Madrid. In 2007, he obtained a position of Associate Professor at the UPM and became part of the Region of Madrid Excellence Group Safety in Automobile Vehicles with special interest in Reduced Mobility Persons. He is currently researcher at the University Institute for Automobile Research. His research interests include Wireless Sensor Networks, intelligent systems, fuzzy logic control, intelligent transport systems, usability, and accessibility. He is the author of more than 100 papers in relevant scientific journals and national and international conferences in the area of ITS and Autonomous Vehicles.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV´18)

INTEGRATION OF COOPERATIVE SERVICES WITH AUTONOMOUS DRIVING

Dr. José E. Naranjo

Madrid, October, 1st

MADRID 2018 ROS

POLITÉCNICA "Ingeniamos el futuro" | INSIA

10TH PLANNING, PERCEPTION AND NAVIGATION FOR INTELLIGENT VEHICLES (PPNIV´18)

POLITÉCNICA "Ingeniamos el futuro" | INSIA | AUTO C-ITS

Integration of Cooperative Services (C-ITS) with Autonomous Driving

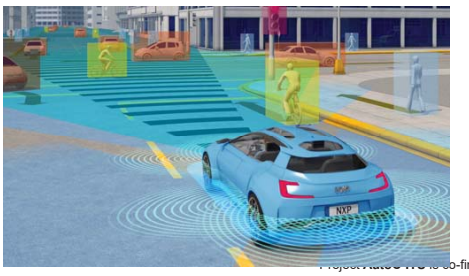
1. CONTEXT OF COOPERATIVE CONNECTED AND AUTOMATED MOBILITY

Connectivity vs Automation

Autonomous vehicles

An autonomous vehicle is a vehicle with the capacity of performing the dynamic driving task. This task includes all of the real-time operational and tactical functions required to operate a vehicle in on-road traffic, excluding the strategic functions such as trip scheduling and selection of destinations and waypoints, and including without limitation:

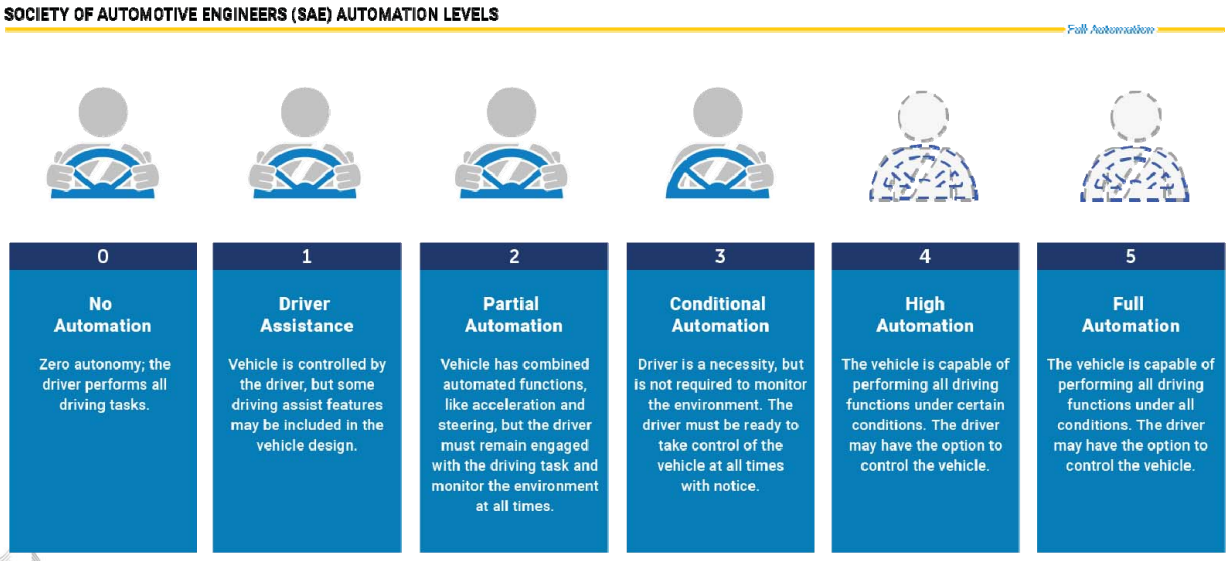
1. Lateral vehicle motion control via steering (operational);
2. Longitudinal vehicle motion control via acceleration and deceleration (operational);
3. Monitoring the driving environment via object and event detection, recognition, classification, and response preparation (operational and tactical)
4. Object and event response execution (operational and tactical);
5. Maneuver planning (tactical); and
6. Enhancing conspicuity via lighting, signaling and gesturing, etc. (tactical).



Connectivity vs Automation

Levels of Driving Automation

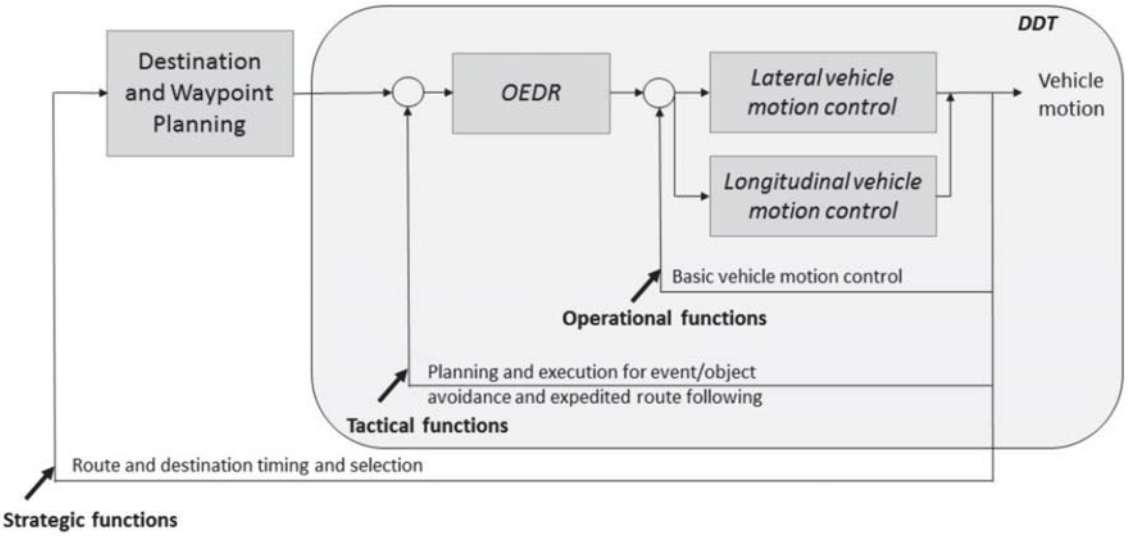
SAE J3016. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles



Connectivity vs Automation

Architecture of an autonomous vehicle

Schematic view of driving task showing the dynamic driving task (DDT)



Connectivity vs Automation

Examples of autonomous vehicles

Volvo

Connectivity vs Automation

Examples of autonomous vehicles

Tesla

Connectivity vs Automation

Examples of autonomous vehicles

Google (Waymo)

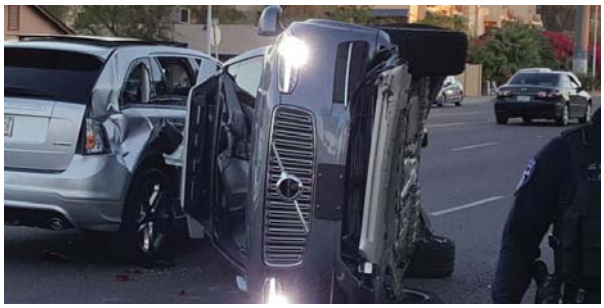
Connectivity vs Automation

Examples of autonomous vehicles

Uber

Connectivity vs Automation

Examples of autonomous vehicles



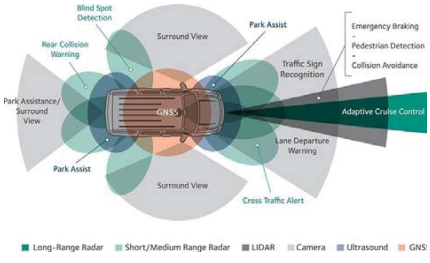
Connectivity vs Automation

Conclusions of the current state of play in autonomous vehicles field

1. There are certain circumstances in which an isolated autonomous vehicle is incapable of responding, limiting itself to the information provided by its own sensors and its driving systems, independently of the:
   - Accuracy of its perception.
   - Intelligence of its auto-pilot.

2. There are certain circumstances that could be solved by the perception and the intelligent pilots but, due the random casuistic, the effort to success in the 100% of the situations is extremely high.

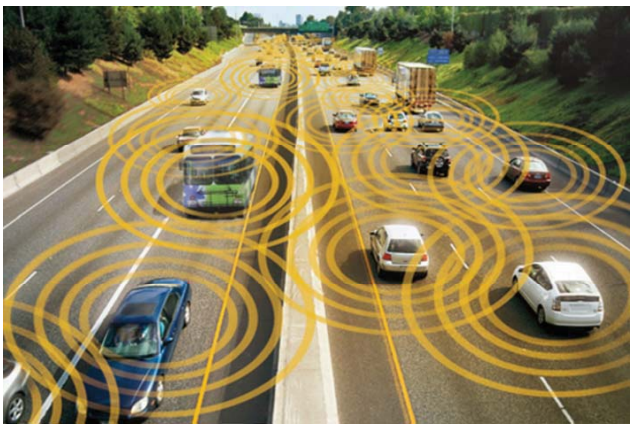
3. There are certain circumstances of medium complexity that increases the workload of the autonomous driving systems and the number of sensors



Connectivity vs Automation

Connected Vehicle

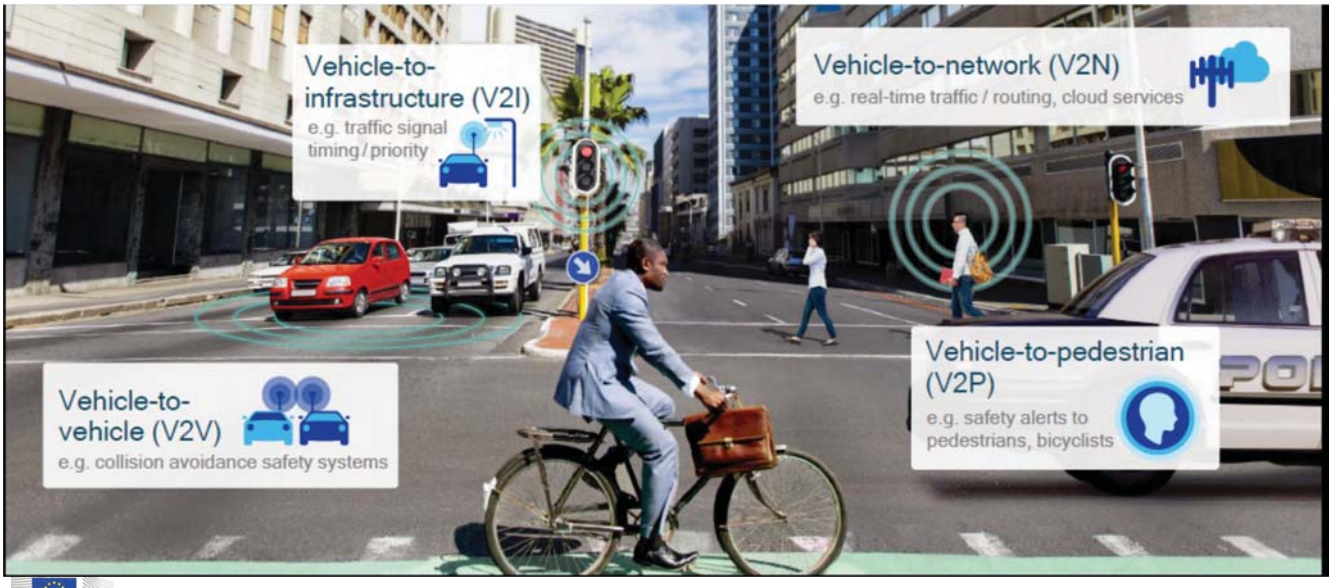
- Connected vehicles are vehicles that use any of a number of different communication technologies to communicate with the driver, other cars on the, roadside infrastructure and the “Cloud”.
- This technology can be used to not only improve vehicle safety, but also to improve vehicle efficiency and commute times.



Connectivity vs Automation

V2X Communications

- Wireless communications have been identified as key technologies for increasing road safety and transportation efficiency.



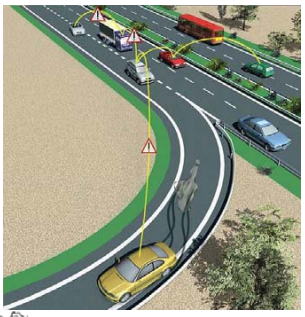
Connectivity vs Automation

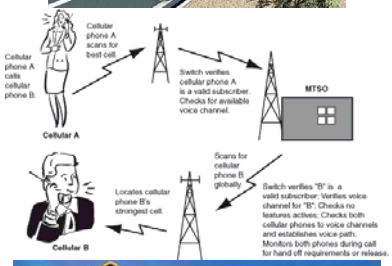
V2X Technology and standardization

Hybrid

- **Short range Communications, based on IEEE 802.11p /ETSI ITS-G5.**
  - Low latencies.
  - Multihop.
  - Geo-Broadcast.
  - No service provider. 5.9 GHz band.
  - Bandwidth: 27 Mbps.
- **Cellular Telephony**
  - Latencies in function of the network load.
  - Network cell schema.
  - Service provider: 3/4 G.
  - Bandwidth: max. 1 Gbps

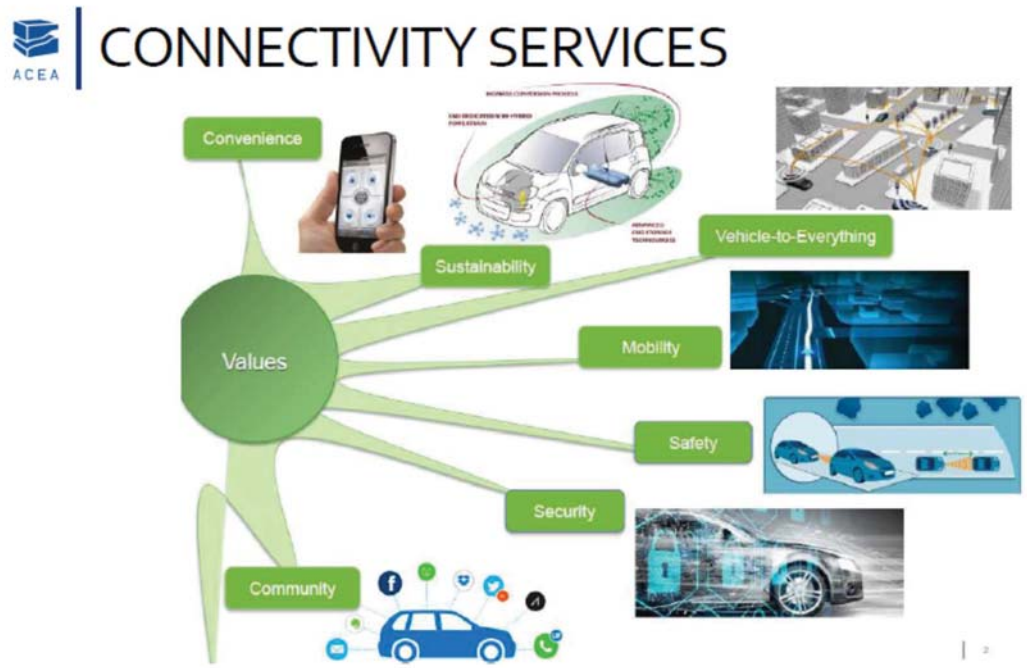
- **5G**
  - Low Latencies.
  - Multihop (Cellular-V2X).
  - Broadcast (Cellular-V2X).
  - Network cell schema; local cell services enabled.
  - Service provider: 5 G.
  - Bandwidth: ∞ Gbps





Connectivity vs Automation

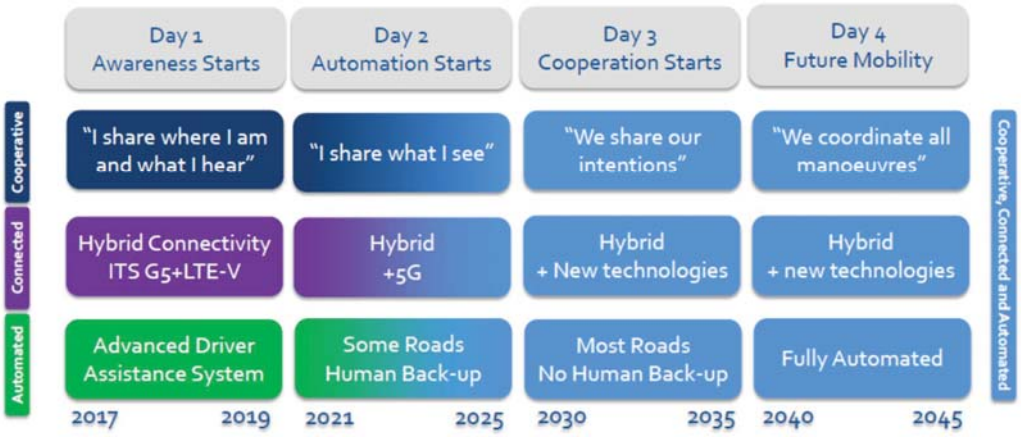
V2X Services



Connectivity vs Automation

V2X Services → Cooperative Systems (C-ITS)

- **European C-ITS Platform**
- Cooperative Intelligent Transport Systems (C-ITS) use technologies that allow road vehicles to communicate with other vehicles, with traffic signals and roadside infrastructure as well as with other road users.



Connectivity vs Automation

V2X Services → C-ITS Day-1, 1.5

These services were chosen on their importance from policy perspectives or potential to answer major societal needs, such as increasing road safety. A further split was introduced based on technical readiness in the short-term (Day 1 vs Day 1.5).

#	Day 1 Services		
1	Emergency electronic brake light	V2V	Safety
2	Emergency vehicle approaching	V2V	Safety
3	Slow or stationary vehicle(s)	V2V	Safety
4	Traffic jam ahead warning	V2V	Safety
5	Hazardous location notification	V2I	Motorway
6	Road works warning	V2I	Motorway
7	Weather conditions	V2I	Motorway
8	In-vehicle signage	V2I	Motorway
9	In-vehicle speed limits	V2I	Motorway
10	Probe vehicle data	V2I	Motorway
11	Shockwave damping	V2I	Motorway
12	GLOSA / Time To Green (TTG)	V2I	Urban
13	Signal violation/Intersection safety	V2I	Urban
14	Traffic signal priority request by designated vehicles	V2I	Urban

#	Day 1.5 Services		
1	Off street parking information	V2I	Parking
2	On street parking information and management	V2I	Parking
3	Park & Ride information	V2I	Parking
4	Information on AFV fuelling & charging stations	V2I	Smart Routing
5	Traffic information and smart routing	V2I	Smart Routing
6	Zone access control for urban areas	V2I	Smart Routing
7	Loading zone management	V2I	Freight
8	Vulnerable road user protection (pedestrians and cyclists)	V2X	VRU
9	Cooperative collision risk warning	V2V	Collision
10	Motorcycle approaching indication	V2V	Collision
11	Wrong way driving	V2I	Wrong Way

Connectivity vs Automation

Convergence between automation and connectivity

Connected

Communicates with other vehicles, nearby infrastructure and ITS backends

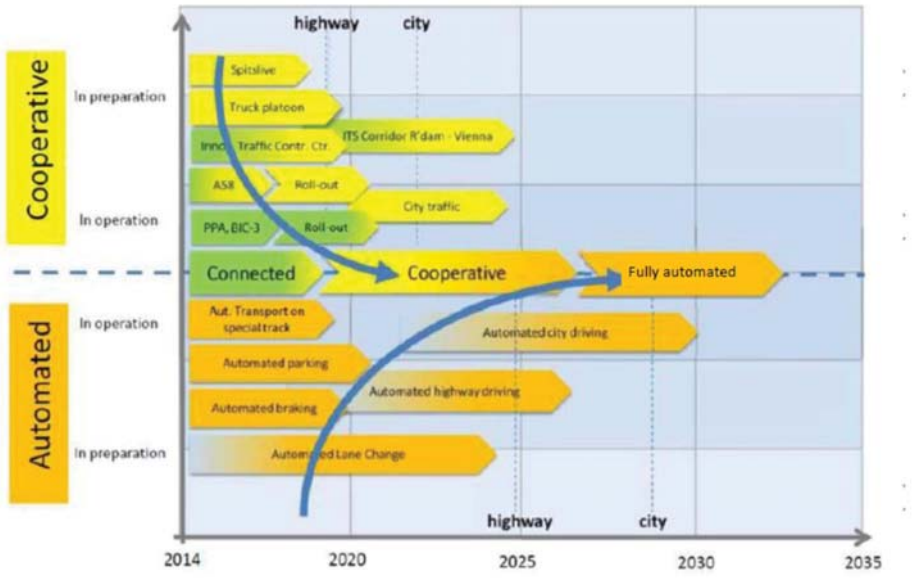
Operates in isolation from other vehicles using internal sensors

Autonomous vehicles

Connected and Autonomous Driving

Connectivity vs Automation

Convergence between automation and connectivity



Connectivity vs Automation

Steps towards Connected and Autonomous Driving

- It is not trivial.
- V2X communications are in continuous evolution.
- Standardization is a key element.
- The evolution of the V2X communications is in parallel with the evolution of autonomous vehicles, but with a shorter time for deployment.
- Direct link with the development and deployment of new generation communications technologies: 5G
- The first step is to take advantage of deployments and technologies made in the field of connected vehicles (C-ITS) to support V2X communications to autonomous vehicles.
- The real implementation of autonomous vehicles without connectivity is almost impossible in real deployments.
- V2X communications technologies are in development, although there are still many elements to solve.
- Cooperative systems can serve as catalysts for the deployment of autonomous and connected driving.
- Unification with the scope of the IOT → Cooperative, Connected and Autonomous Mobility (CCAM).

Connectivity vs Automation

Declaration of Amsterdam

- Declaration of Amsterdam
- 14 April 2016
- Signed by the transport ministers of all 28 EU member states

https://english.eu2016.nl/binaries/eu2016-en/documents/publications/2016/04/14/declaration-of-amsterdam/2016-04-08-declaration-of-amsterdam-final-format-3.pdf

Connectivity vs Automation

Declaration of Amsterdam

Highlights (objectives)

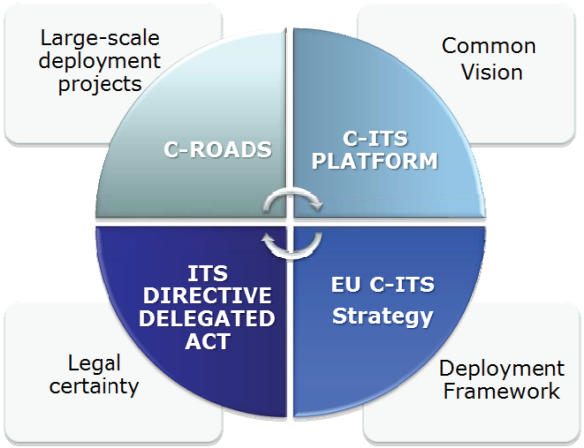
- to work towards a coherent European framework for the deployment of interoperable connected and automated driving, which should be available, if possible, by 2019;
- to bring together developments of connected and automated driving in order to reach their full potential to improve road safety, human health, traffic flows, and to reduce the environmental impact of road transport;
- to adopt a “learning by experience” approach, including, where possible, crossborder cooperation, sharing and expanding knowledge on connected and automated driving and to develop practical guidelines to ensure interoperability of systems and services;
- to support further innovation in connected and automated vehicle technologies to strengthen the global market position of European industry; and
- to ensure data protection and privacy.

Connectivity vs Automation

V2X European projects

The C-Roads Platform is a joint initiative of European Member States and road operators for testing and implementing C-ITS services in light of cross-border harmonization and interoperability.

C-ROADS → National Mirrors → C-ROADS SPAIN



2. EUROPEAN PROJECT AUTOCITS IN THE CONTEXT OF COOPERATIVE CONNECTED AND AUTOMATED MOBILITY

AUTOCITS - In a nutshell

*"AUTOCITS aims to **contribute to the deployment of C-ITS in Europe** and to boost the role of C-ITS as catalyst for the implementation of autonomous driving"*

C-ITS: Intelligent Transport Systems (ITS) where ITS stations (vehicles, roadside equipment, traffic control centers and personal devices) communicate and share information

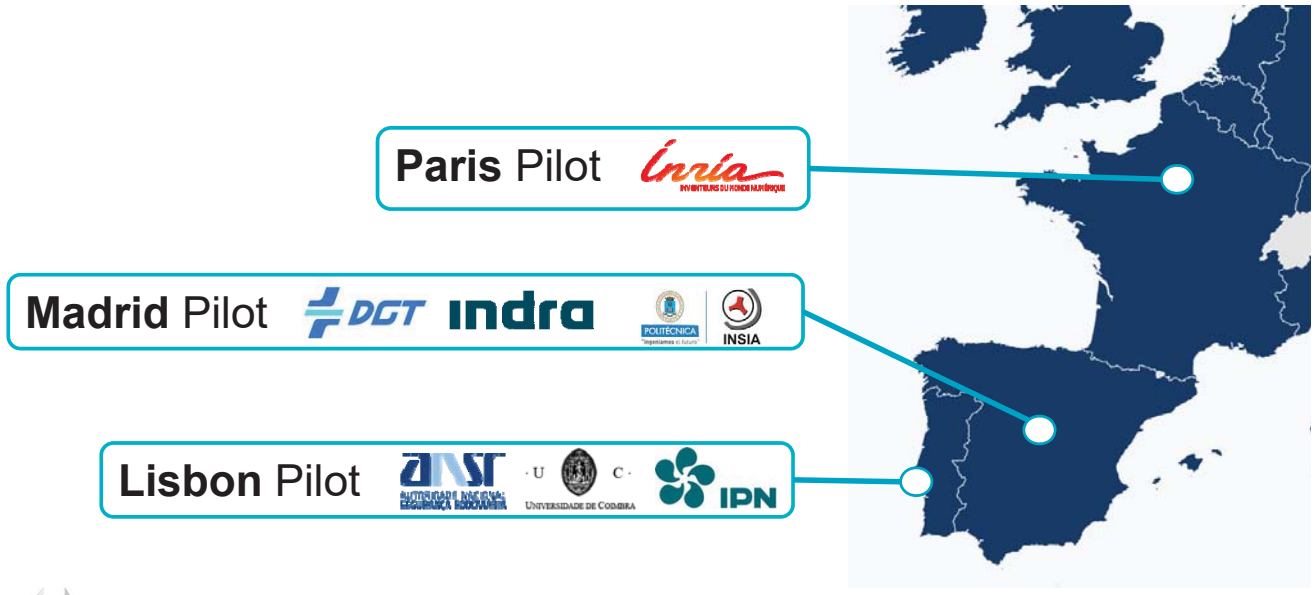
CAD – Connected & Autonomous Driving take advantage of a variety of techniques to detect their surroundings and advanced control systems to interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage

Regulatory Framework

European Transport Network – Atlantic Corridor

AUTOCITS: Partners & Figures

Programme: Connected Europe Facility
Starting date: 01-11-2016
Ending Date: 31-03-2019
Duration: 29 months
Call: CEF- 2015
Budget : 2,606,550 €
Coordinator: INDRA
Funding: 50%



AUTOCITS - Objetives

Study on the current National, European and International legal framework for autonomous driving

Pilot C-ITS services for autonomous vehicles (AVs) under the applicable traffic regulation

AUTOCITS

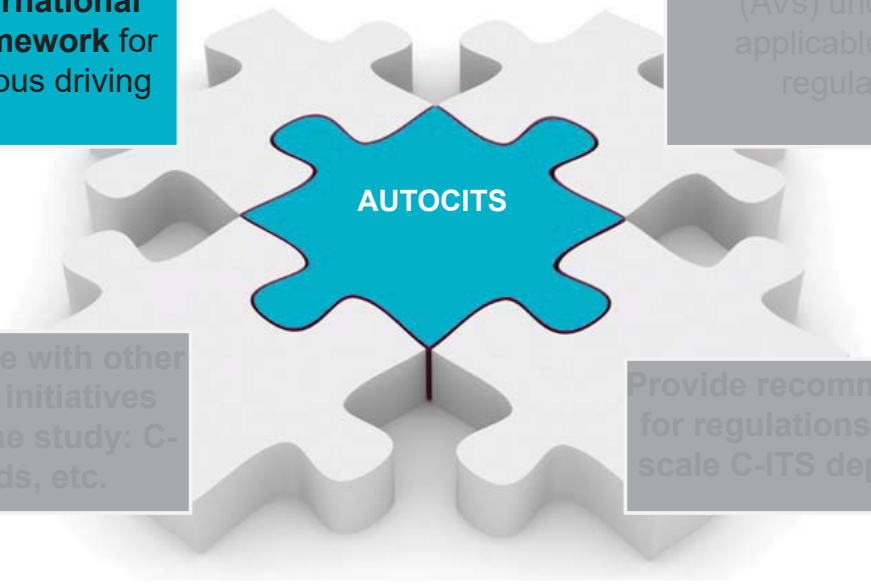
Cooperate with other current initiatives during the study: C-Roads, etc.

Provide recommendations for regulations and large scale C-ITS deployments

AUTOCITS - Objetives

Study on the current National, European and International legal framework for autonomous driving

Pilot C-ITS services for autonomous vehicles (AVs) under the applicable traffic regulation



Cooperate with other current initiatives during the study: C-Roads, etc.

Provide recommendations for regulations and large scale C-ITS deployments

Regulation study in AUTOCITS (Study)

Study of the national and European regulatory frameworks for the deployment of the Autonomous Driving

United States of America, Japan, Singapore, South Korea , China, Australia, etc.

Making propositions and recommendations for regulation and legal framework

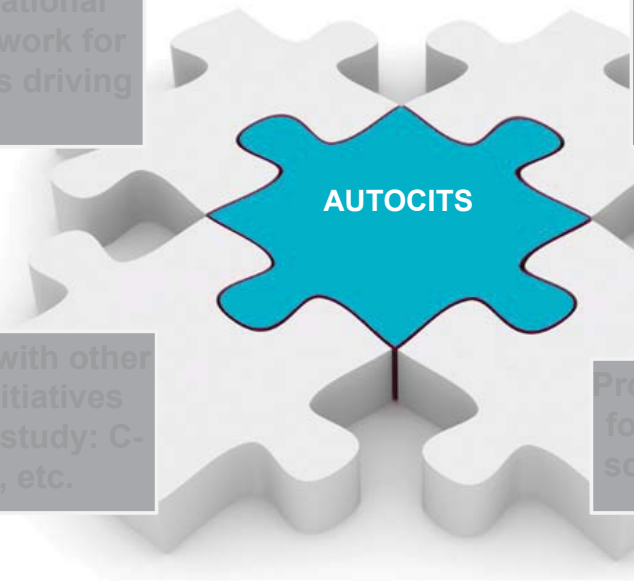
Some of the **aspects under study** are:

Alignment with Vienna Convention
Normative on driving
Testing Legislation
Vehicle certification (individual vehicles, mass production)
Laws to be modified
Changes on SAE 3-5 already initiated/foreseen

AUTOCITS - Objetives

Study on the current National, European and International legal framework for autonomous driving

Pilot C-ITS services for autonomous vehicles (AVs) under the applicable traffic regulation



Cooperate with other current initiatives during the study: C-Roads, etc.

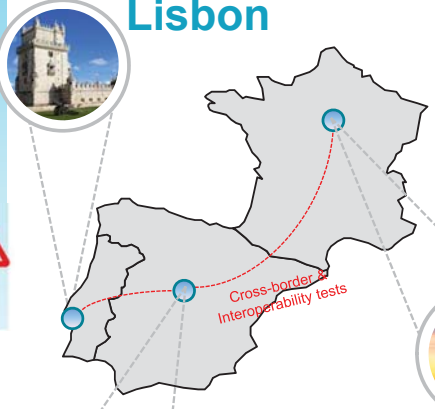
Provide recommendations for regulations and large scale C-ITS deployments

AUTOCITS Pilots

3 Pilots in the Atlantic Corridor

Location:
A9 – CREL Circular Regional Externa de Lisboa
Day 1 C-ITS Services:
•Slow or stationary vehicle & traffic ahead warning
•Weather conditions
•Other hazardous notifications
Test vehicles
1 autonomous vehicle
1 instrumented vehicle
1 autonomous shuttles

Lisbon



Location:
The highway A13
Day 1 C-ITS Service:
•Slow or stationary vehicle & traffic ahead warning
•Weather conditions
•Other Hazardous notifications
Test vehicles
4 connected vehicles
1 autonomous vehicle

Paris

Madrid

Location:
The HOV Lane located between the M30 and M40
Day 1 C-ITS Services:
•Slow or stationary vehicle & traffic ahead warning
•Road works warning
•Weather conditions
Test vehicles
4 instrumented and connected vehicles
2 autonomous vehicles

Pilot Overview - Spain

Road: A6 Autovía del Noroeste, stretch between M30 and M40, Reversible high occupancy lane
Length: 10 kms, **15 RSUs** have been installed

Traffic conditions
- More than 20.000 vehicles/day
- Close to traffic: controlled tests
- Open to traffic: private vehicles and public collective transport (bus)

Vehicles involved
- **Autonomous vehicles:** 2 vehicles
- **Connected vehicles:** 4 vehicles

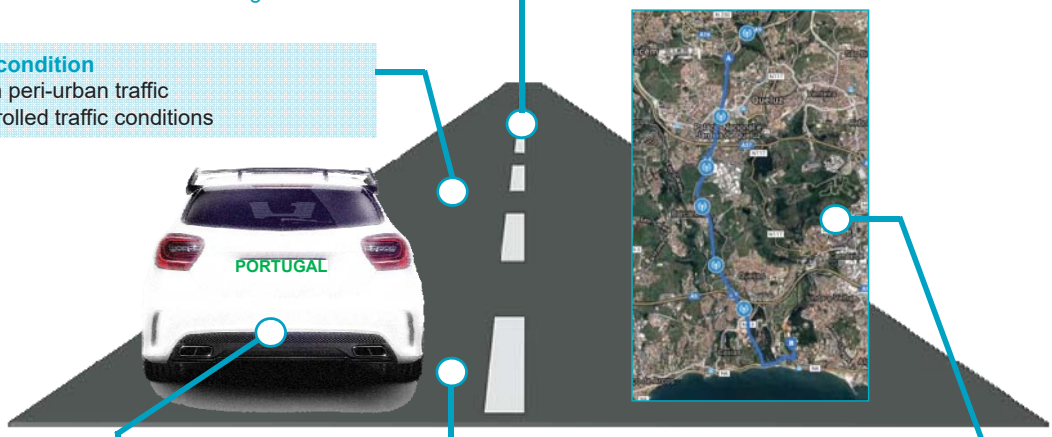

C-ITS Day 1 services
- **Service 1:** Road Works information service
- **Service 2:** Weather information service
- **Service 3:** Traffic ahead service

Communication Channel
- **ITS G5**

Pilot Overview - Portugal

Roads
1) A9-CREL Between (Cirular Regional Enterior de Lisboa) and a national
Length: 7 kms , **5 RSUs** have been installed
2) road connecting A9 and Faculty of Human Kinetics
Length: 1kms

Traffic condition
1) Open peri-urban traffic
2) Controlled traffic conditions



Vehicles involved
- **Autonomous vehicles**: 1 vehicles
- **Autonomous shuttle:** 1 vehicle
- **Connected vehicles:** 1 vehicles

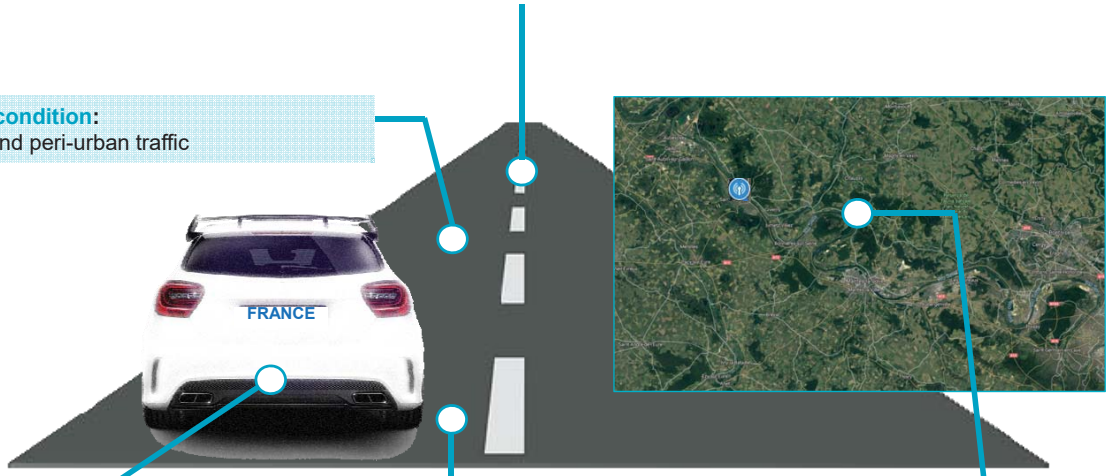

C-ITS Day 1 services
- **Service 1:** Notification of slow or stationary vehicles
- **Service 2:** Weather information service
- **Service 3:** Other hazardous notifications

Communication Channel
- **ITS G5**

Pilot Overview - France

Road: Peri-Urban A13 Highway entrance to Paris
Number of RSUs: 1 **RSU** has been installed

Traffic condition:
Urban and peri-urban traffic



Vehicles involved
- **Autonomous vehicles:** C1 Evie
- **Connected vehicles:** 4 C3 vehicles

C-ITS Day 1 services
- **Service 1:** hazardous location notification
- **Service 2:** contextual speed adapting
- **Service 3:** traffic scheduling assist

Communication Channel
- **ITS G5**

Interoperability and Cross-border tests

Initial interoperability lab tests: (MADRID, February '18)

Test Infrastructure:
- INSIA Lab Equipment
- V2X Equipment from 5 manufacturers involved in all pilots

Test Objective: Validating compatibility on:
- Frequency channel
- Physical level compatibility
- Sending/Reception of CAM/DEMN messages

Test Results:
- Total compatibility at physical level.
- Frequency channel stablished in 5.900 GHz.
- Stable geo-networking version 0.1.
- Success in interoperability. Sending & reception of CAM/DENM messages.
- The ITS station of the 5 manufacturers are interoperable at the AUTOCITS premises.

Initial cross-border tests: (LISBON, July '18):

Test infrastructure:
- Two connected vehicles
- V2X equipment from 3 manufacturers

Test Objectives:
- Ensure interoperability of one C-ITS Service (Traffic ahead warning)

Test Results:
- Timestamp origin of times is the same for all teams and are synchronized
- All fields of DEMN messages should be filled to be detected as DEMN
- MAC identification should be unique for each RSU
- Number of hops should be defined in order to forward of messages

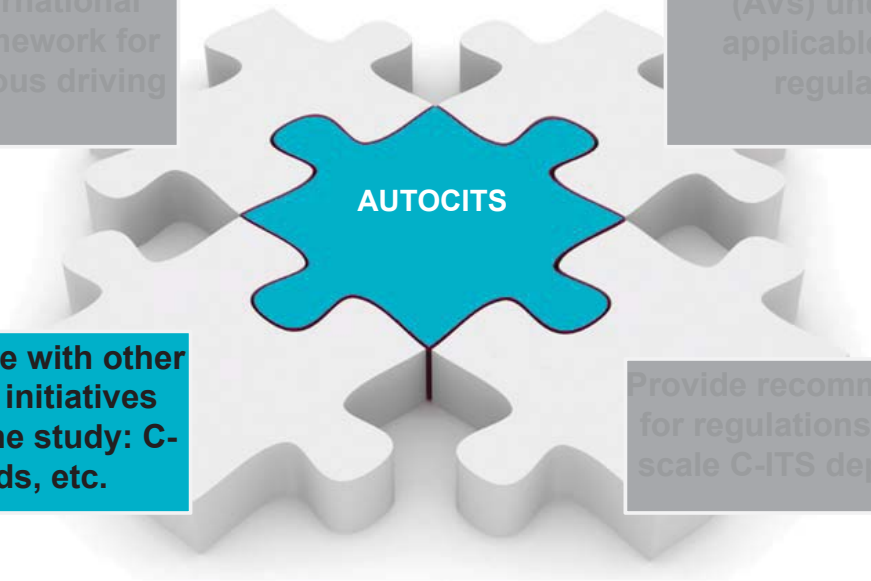
Initial Conclusions:
- **Synchronization** of the time zone is needed
- The equipment must all work in the **same frequency**
- Same **versions of geonetworking** protocols must be implemented

Next Cross border tests
- **Lisbon** – 15th / 19th Of October 2018
- Additional C-ITS services
- Autonomous Vehicles

AUTOCITS - Objetives

Study on the current National, European and International legal framework for autonomous driving

Pilot C-ITS services for autonomous vehicles (AVs) under the applicable traffic regulation



Cooperate with other current initiatives during the study: C-Roads, etc.

Provide recommendations for regulations and large scale C-ITS deployments

Contribution to/from the C-ROADS Platform

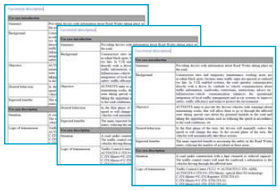
WG2 Technical Aspects/ WG3 Evaluation methodology

TF2 Service Harmonisation
TF3 Infrastructure Communication
TF4 Hybrid Communication
TF5 Cross border Validation

EXPECTED CONTRIBUTION TO THE PLATFORM

AUTOCITS C-ITS specifications for Harmonised C-ITS specifications

Road Weather warning
Roadworks warning
Traffic ahead warning



- Implementation of services
- Provision of Communication model used
- Results of cross-border validation tests
- Results from pilots assessment and evaluation

EXPECTED CONTRIBUTION FROM PLATFORM

- Harmonised C-ITS specifications
- Evaluation and assessment plan
- Use of service standardisation
- Adpotions of Infrastructure Communication model
- Application of Hybrid Communications vision
- Cross border Validation tests
- Strategy for assessment and evaluation

Contribution to/from the C-ROADS Spain

AUTOCITS - Objetives

Study on the current National, European and International legal framework for autonomous driving

Pilot C-ITS services for autonomous vehicles (AVs) under the applicable traffic regulation

AUTOCITS

Cooperate with other current initiatives during the study: C-Roads, etc.

Provide recommendations for regulations and large scale C-ITS deployments

Workshops

1st AUTOCITS WORKSHOP
MADRID, Nov 23rd 2017

2nd AUTOCITS WORKSHOP
PARIS, May 10th 2017

3rd AUTOCITS WORKSHOP
Lisbon, October 10th 2017

1st INTERNATIONAL WORKSHOP
Cologne, 5th July 2017

4th AUTOCITS WORKSHOP
Madrid February 2018

2nd INTERNATIONAL WORKSHOP
Vienna, 17th April 2018

5th AUTOCITS WORKSHOP
PARIS, Dec 11th 2018

6th AUTOCITS WORKSHOP
Lisbon, February 2019

FINAL AUTOCITS WORKSHOP
Madrid, March 2018

Pilot Deployment

Connected Vehicles

Autonomous Driving Pilot Deployment

Autonomous Vehicles

AUTO C-ITS

Regulation Study for Interoperability in the Adoption the Autonomous Driving in European Urban Nodes

Project **AutoC-ITS** is co-financed by the European Union's Connecting Europe Facility (CEF)

Co-financed by the European Union
Connecting Europe Facility

Thanks for your attention

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session V

Control, Planning

Title: Multi-Sensor-Based Predictive Control for Autonomous Backward Perpendicular and Diagonal Parking
Authors: D. Perez-Morales, O. Kermorgant , S. Domınguez-Quijada and P. Martinet

Title: Towards Uncertainty-Aware Path Planning for Navigation on Road Networks Using Augmented MDPs
Authors: L. Nardi, C. Stachniss

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Multi-Sensor-Based Predictive Control for Autonomous Backward Perpendicular and Diagonal Parking

David Pérez-Morales[1], Olivier Kermorgant[2], Salvador Domínguez-Quijada[3] and Philippe Martinet[4]

***Abstract*— This paper explores the feasibility of a Multi-Sensor-Based Predictive Control (MSBPC) approach for addressing backward nonparallel (perpendicular and diagonal) parking problems of car-like vehicles as an alternative to more classical (e.g. path planning based) approaches. The results of a few individual cases are presented to illustrate the behavior and performance of the proposed approach as well as results from exhaustive simulations to assess its convergence and stability. Indeed, preliminary results are encouraging, showing that the vehicle is able to park successfully from virtually any sensible initial position.**

I. INTRODUCTION

Even though the research on autonomous parking started more than 20 years ago, leading to a quite extensive literature [1] and in spite of the fact that the automobile industry has already started to roll out some commercial implementations of active parking assistants capable of actively controlling acceleration, braking and steering [2], the research interest in the topic remains strong. This is, partially at least, due to the ever-growing size of many cities around the world, leading to an increment in the number of automobiles in the streets and thus causing parking to become an increasingly difficult and dangerous task.

Path planning approaches have been heavily investigated in recent years. Among the different planning techniques it is possible to distinguish between geometric approaches, with either constant turning radius [3], [4] using saturated feedback controllers, or continuous-curvature planning using clothoids [5], [6]; heuristic approaches [7] and machine learning techniques [8].

A well-known drawback of path planning is that it is necessary to have knowledge about the free and occupied space of the whole environment beforehand if online replanning is not feasible, potentially leading to costly infrastructure requirement. Moreover, it is known that path planning algorithms that consider some kind of space exploration step (such as A*, RRT, etc.) have to make a compromise between computation time and exploration's completeness. Furthermore, the tracking performance of a given path is highly dependent on the localization performance which might get degraded on certain environments (e.g. underground parking lots without any special infrastructure) or after a few maneuvers leading to non-negligible differences between the planned path and the performed one [5], [6].

An interesting alternative is the use of a sensor-based control approach. It has been proven to be valid for navigation [9], dynamic obstacle avoidance [10] and for parking applications [11], [12]. It should be noted that an important limitation of a purely sensor-based control approach is the possibility of getting trapped in local minima – i.e. if the car is not able to park in one maneuver from the initial pose then the parking maneuver won't be successful.

A. Reasoning and contribution

A natural goal for a human driver when parking would be to try to make the vehicle's longitudinal axis to be collinear to the main axis of the parking spot (i.e. to be centered lateral-wise) and finish the maneuver at a certain distance from the rear boundary of the parking spot while avoiding collision with surrounding obstacles during the whole maneuver.

Assuming that the vehicle is capable of perceiving surrounding free parking spots, it is possible to park without any path planning using a Multi-Sensor-Based Predictive Control (MSBPC) approach by minimizing the error between the current value of a certain set of sensor features (i.e. a line collinear to the parking spot's main axis and another collinear to the rear boundary of the parking spot) and its desired value while avoiding collision by imposing certain constraints on another set of sensor features (lines defining the boundaries of the parking spot, points at the corners of said spot, etc.). It is worth noting that, since the presented approach is based on the features perceived at each time instant and a certain desired fixed value for each feature, no localization is inherently required for it to be stable in spite of the prediction step considered.

The contribution of this paper is the exploration of a MSBPC approach for backward perpendicular and diagonal parking, being able now to park with multiple maneuvers. It should be noted that, in order to decouple the performance of the controller from the perception, the sensory data is generated virtually and assumed to be available all the time.

B. Contents of the paper

In the next section the kinematic model of the vehicle and the multi-sensor modeling are presented. Section III describes the interaction model allowing to formalize the parking tasks and the constraints for collision avoidance. Afterwards, the controller is presented in Section IV. The

[1] David Pérez-Morales, [2]Olivier Kermorgant and [3]Salvador Domínguez-Quijada are with LS2N, Laboratoire des Sciences du Numérique de Nantes, École Centrale de Nantes, 1 rue de la Noë, 44321 Nantes, France

[4] Philippe Martinet is with INRIA Sophia Antipolis, 2004 Route des Lucioles, 06902 Valbonne, France and École Centrale de Nantes - LS2N, 1 rue de la Noë, 44321 Nantes, France

[1] David.Perez-Morales@eleves.ec-nantes.fr
[2] Olivier.Kermorgant@ec-nantes.fr
[3] Salvador.DominguezQuijada@ls2n.fr
[4] Philippe.Martinet@inria.fr

obtained results are presented in Section V: a few cases in two different simulation environments are presented as well as exhaustive simulations results for assessing the convergence performance of the presented approach for the two different types of parking maneuvers addressed are shown. Finally, some conclusions are given in Section VI.

II. MODELING AND NOTATION

Given that parking maneuvers are low-speed motions, a kinematic model can be considered as accurate enough.

A. Car-like robot model and notation

The considered kinematic model is a car with rear-wheel driving:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/l_{wb} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{\phi}, \tag{1}$$

where v and $\dot{\phi}$ are the longitudinal and steering velocities.

The point M is located at the mid-distance between the passive fixed wheels (rear) axle and the distance between the rear and the front axle is described by l_{wb}. The generalized coordinates are $\mathbf{q} = [x, y, \theta, \phi]^\mathsf{T}$ where x and y are the Cartesian coordinates of the point M, θ is the orientation of the platform with respect to the x_0 axis and the steering angle of the steerable wheel(s) is denoted by ϕ (Fig. 1a).

The turning radius ρ_m around the instantaneous center of rotation (ICR) can be defined as:

$$\rho_m = \frac{l_{wb}}{\tan\phi} \tag{2}$$

The vehicle used for experimentation and simulation, represented by its bounding rectangle in Fig. 1a, is a Renault ZOE (Fig. 1b). Its relevant dimensional parameters are presented in Table I.

TABLE I
DIMENSIONAL VEHICLE PARAMETERS

Parameters	*Notation*	*Value*
Wheelbase: Distance between the front and rear wheel axles	l_{wb}	2.588 m
Rear overhang: Distance between the rear wheel axle and the rear bumper	l_{ro}	0.657 m
Total length of the vehicle	l_{ve}	4.084 m
Total width of the vehicle	w_{ve}	1.945 m

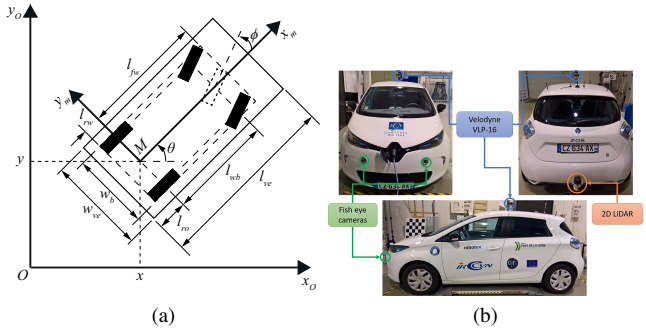
B. Multi-sensor modeling

The considered multi-sensor modeling is recalled in this subsection.

1) Kinematic model: Let us consider a robotic system equipped with k sensors (Fig. 2) that provide data about the environment. Each sensor S_i gives a signal (sensor feature) $\mathbf{s}_i$ of dimension $\mathbb{d}_i$ with $\sum_{i=1}^{k} \mathbb{d}_i = \mathbb{d}$.

In a static environment, the sensor feature derivative can be expressed as follows:

$$\dot{\mathbf{s}}_i = \breve{\mathbf{L}}_i \breve{\mathbf{v}}_i = \breve{\mathbf{L}}_i {}^i\breve{\mathbf{T}}_m \breve{\mathbf{v}}_m \tag{3}$$



(a) (b)

Fig. 1. (a) Kinematic model diagram for a car-like rear-wheel driving robot. (b) Robotized Renault ZOE used for real experimentation

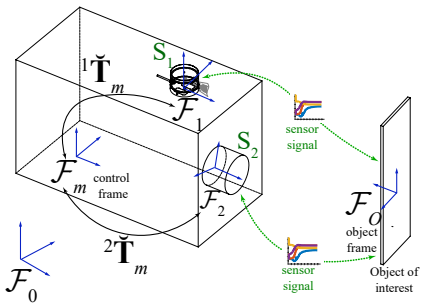


Fig. 2. Multi-sensor model

where $\breve{\mathbf{L}}_i$ is the interaction matrix [13] of $\mathbf{s}_i$ ($\dim(\breve{\mathbf{L}}_i) = \mathbb{d}_i \times 6$) and ${}^i\breve{\mathbf{T}}_m$ is the 3D screw transformation matrix that allows expressing the sensor twist $\breve{\mathbf{v}}_i$ (which is expressed in its corresponding frame $\mathcal{F}_i$) with respect to the robot twist $\breve{\mathbf{v}}_m$ (expressed in the control frame $\mathcal{F}_m$).

Denoting $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_k)$ the $\mathbb{d}$-dimensional signal of the multi-sensor system, the signal variation over time can be linked to the moving vehicle twist:

$$\dot{\mathbf{s}} = \breve{\mathbf{L}}_\mathbf{s} \breve{\mathbf{v}}_m \tag{4}$$

with:

$$\breve{\mathbf{L}}_\mathbf{s} = \breve{\mathbf{L}} \breve{\mathbf{T}}_m \tag{5}$$

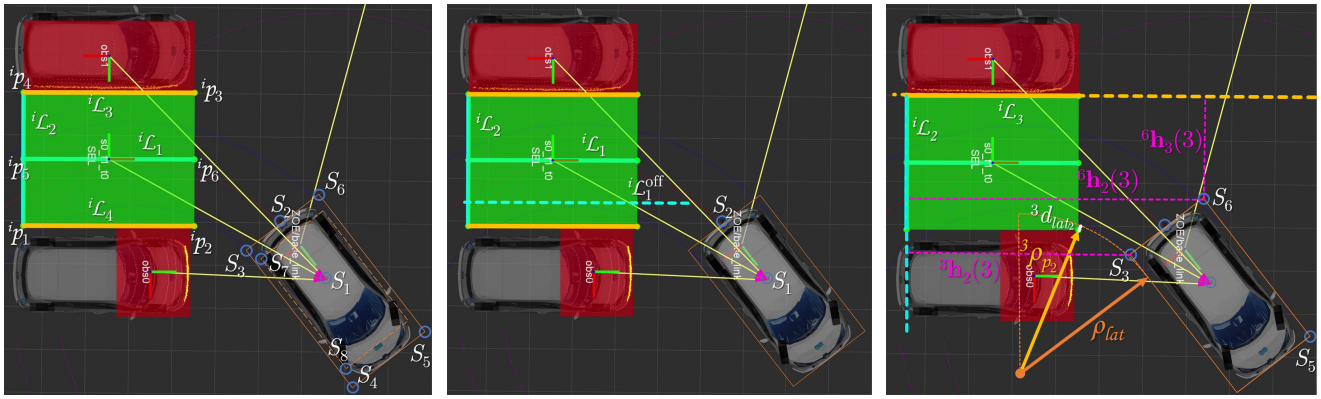
where $\breve{\mathbf{L}}$ and $\breve{\mathbf{T}}_m$ are obtained by concatenating either diagonally or vertically, respectively, matrices $\breve{\mathbf{L}}_i$ and ${}^i\breve{\mathbf{T}}_m$ $\forall\, i \in [1 \ \ldots \ k]$.

Planar world assumption: Assuming that the vehicle to which the sensors are rigidly attached evolves in a plane and that the sensors and vehicle have vertical parallel z axes, all the twists are reduced to $[v_{x_i}, v_{y_i}, \dot{\theta}_i]^\mathsf{T}$ hence the reduced forms $\breve{\mathbf{L}}$, $\breve{\mathbf{L}}_\mathbf{s}$, $\breve{\mathbf{L}}_i$, $\breve{\mathbf{v}}_m$ and ${}^i\breve{\mathbf{T}}_m$ of, respectively, $\breve{\mathbf{L}}$, $\breve{\mathbf{L}}_\mathbf{s}$, $\breve{\mathbf{L}}_i$, $\breve{\mathbf{v}}_m$ and ${}^i\breve{\mathbf{T}}_m$ are considered.

$\breve{\mathbf{L}}_i$ is of dimension $\mathbb{d}_i \times 3$, $\breve{\mathbf{v}}_m = [v_{x_m}, v_{y_m}, \dot{\theta}_m]^\mathsf{T}$ and ${}^i\breve{\mathbf{T}}_m$ is defined as:

$${}^i\breve{\mathbf{T}}_m = \begin{bmatrix} \cos({}^m\theta_i) & \sin({}^m\theta_i) & x_i \sin({}^m\theta_i) - y_i \cos({}^m\theta_i) \\ -\sin({}^m\theta_i) & \cos({}^m\theta_i) & x_i \cos({}^m\theta_i) + y_i \sin({}^m\theta_i) \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where ${}^m\mathbf{t}_i = [x_i, y_i]^\mathsf{T}$ and ${}^m\theta_i$ are, respectively, the position and orientation of S_i (frame $\mathcal{F}_i$) with respect to $\mathcal{F}_m$ expressed in $\mathcal{F}_m$.



(a) Sensors' configuration and sensor features (b) Task features (c) Constrainted features illustration

Fig. 3. (a) General sensors' configuration and sensor features. (b) Features considered for the parking task. (c) Example of the constrained sensor features

Furthermore, since in the considered model the control frame $\mathcal{F}_m$ is attached to the vehicle's rear axis with origin at the point M (Fig. 1a), it is not possible to generate a velocity along y_m on the vehicle's frame and assuming that there is no slipping nor skidding (i.e. $v_{y_m} = 0$), the robot twist $\check{\mathbf{v}}_m$ can be further reduced to:

$$\mathbf{v}_m = [v_{x_m}, \dot{\theta}_m]^\mathsf{T} \tag{7}$$

with $v_{x_m} = v$ and $\dot{\theta}_m = \dot{\theta}$ according to the model (1), thus it is possible to write:

$$\dot{\mathbf{s}} = \mathbf{L_s}\, \mathbf{v}_m \tag{8}$$

where $\mathbf{L_s}$ is composed of the first and third columns of $\check{\mathbf{L}}_\mathbf{s}$.

III. INTERACTION MODEL

For the interaction model, we rely on the perception of several lines $\mathcal{L}_j$ and points from several (virtual) sensors placed at in convenient frames in order to simplify the sensor features definitions and their interaction matrices. The usefulness of virtual sensors can be exemplified as follows: if the car is parking into perpendicular spot with a backward motion (Fig. 3a), the risk of collision with the obstacle on the left is the highest for the car's rear left corner, therefore it would be convenient to have a virtual sensor (S_6) placed on said corner to measure directly the distance to left boundary ($^6\mathcal{L}_3$) of the parking spot.

The sensor's placement can be seen in Fig. 3a. S_1 corresponds to the VLP-16 while S_2 to the 2D LiDAR placed on the rear (LMS151). S_3 to S_6 are placed on the corners of the car's bounding rectangle and have the same orientation as the control frame.

As it can be seen in Fig. 3a, points p_1 to p_4 correspond to the corners of the parking spot while p_5 and p_6 are, respectively, the midpoints between (p_1, p_4) and (p_2, p_3). $\mathcal{L}_1$ is a line that passes through p_5 and p_6, i.e. it passes through the center of the parking spot. $\mathcal{L}_2$ is a line that passes through p_1 and p_4 thus corresponding to the rear boundary of the parking spot. $\mathcal{L}_3$ is a line that passes through p_3 and p_4. All the lines are parametrized using normalized Plücker coordinates.

A. Line parametrization

Given two distinct 3D points ^ip_f and ^ip_g in homogeneous coordinates, with

$$^ip_f = [^iX_f, {}^iY_f, {}^iZ_f, {}^iW_f]^\mathsf{T} \tag{9a}$$

$$^ip_g = [^iX_g, {}^iY_g, {}^iZ_g, {}^iW_g]^\mathsf{T}, \tag{9b}$$

a line passing through them can be represented using normalized Plücker coordinates as a couple of 3-vectors [14]:

$$^i\mathcal{L}_j = [^i\underline{\mathbf{u}}_j, {}^i\mathbf{h}_j]^\mathsf{T} \tag{10}$$

where $^i\underline{\mathbf{u}}_j = {}^i\mathbf{u}_j / ||{}^i\mathbf{u}_j||$ (with $^i\mathbf{u}_j \neq 0$) describes the orientation of the line and $^i\mathbf{h}_j = {}^i\mathbf{r}_j / ||{}^i\mathbf{u}_j||$ where $^i\mathbf{r}_j$ encodes the plane containing the line and the origin (*interpretation plane*) and the distance from the origin to the line. The two 3-vectors $^i\mathbf{u}_j$ and $^i\mathbf{r}_j$ are defined as [15]:

$$^i\mathbf{u}_j^\mathsf{T} = {}^iW_f[^iX_g, {}^iY_g, {}^iZ_g] - {}^iW_g[^iX_f, {}^iY_f, {}^iZ_f] \tag{11a}$$

$$^i\mathbf{r}_j^\mathsf{T} = [^iX_f, {}^iY_f, {}^iZ_f] \times [^iX_g, {}^iY_g, {}^iZ_g] \tag{11b}$$

Due to the planar world assumption considered in this paper, the third element of $^i\underline{\mathbf{u}}_j$ and the first and second elements of $^i\mathbf{h}_j$ are equal to zero, i.e. $^i\underline{\mathbf{u}}_j(3) = {}^i\mathbf{h}_j(1) = {}^i\mathbf{h}_j(2) = 0$, therefore the sensor signal $\mathbf{s}_{i_{\mathcal{L}_j}}$ and interaction matrix $\check{\mathbf{L}}_{i_{\mathcal{L}_j}}$ for the line $^i\mathcal{L}_j$ observed by S_i are defined respectively as:

$$\mathbf{s}_{i_{\mathcal{L}_j}} = \left[{}^i\underline{\mathbf{u}}_j(1), {}^i\underline{\mathbf{u}}_j(2), {}^i\mathbf{h}_j(3)\right]^\mathsf{T} \tag{12}$$

$$\check{\mathbf{L}}_{i_{\mathcal{L}_j}} = \begin{bmatrix} 0 & 0 & {}^i\underline{\mathbf{u}}_j(2) \\ 0 & 0 & -{}^i\underline{\mathbf{u}}_j(1) \\ -{}^i\underline{\mathbf{u}}_j(2) & {}^i\underline{\mathbf{u}}_j(1) & 0 \end{bmatrix} \tag{13}$$

B. Task sensor features

The set of task sensor features $\mathbf{s}^t$ is defined as:

$$\mathbf{s}^t = [s_1^t, \ldots, s_9^t]^\mathsf{T} = [\mathbf{s}_1^t, \mathbf{s}_2^t]^\mathsf{T} = [\mathbf{s}_{1_{\mathcal{L}_1^{\text{off}}}}, \mathbf{s}_{2_{\mathcal{L}_1}}, \mathbf{s}_{2_{\mathcal{L}_2}}]^\mathsf{T}, \tag{14}$$

where $^1\mathcal{L}_1^{\text{off}}$ is simply $^1\mathcal{L}_1$ with an offset to the right with respect to the parking spot (Fig. 3b).

The idea behind considering $\mathbf{s}_{1_{\mathcal{L}_1^{\text{off}}}}$ in addition to $\mathbf{s}_2^t$ as part of the set of task sensor features is to have some features that will pull the vehicle out of the parking spot with a forward motion, like a human driver would likely do, in order to escape from local minima therefore being able to park with multiple maneuvers.

The interaction matrix $\check{\mathbf{L}}_1^t$ for the features observed by S_1 is computed at each iteration and is defined by (13) while, for the features observed by S_2, the corresponding interaction matrix $\check{\mathbf{L}}_2^t$ is computed by a 2nd order approximation [16] of the form:

$$\check{\mathbf{L}}^t = \frac{\check{\mathbf{L}}_{\mathcal{L}} + \check{\mathbf{L}}_{\mathcal{L}}^*}{2} \tag{15}$$

where $\check{\mathbf{L}}_{\mathcal{L}} = [\check{\mathbf{L}}_{i_{\mathcal{L}_1}}, \check{\mathbf{L}}_{i_{\mathcal{L}_2}}]^\mathsf{T}$ and $\check{\mathbf{L}}_{\mathcal{L}}^*$ is equal to the value of $\check{\mathbf{L}}_{\mathcal{L}}$ at the desired pose.

Considering the definition of ${}^i\mathcal{L}_1$ and ${}^i\mathcal{L}_2$, a sensible choice would be for ${}^i\mathcal{L}_1^*$ to be collinear with the vehicle's longitudinal axis (x_m-axis) and ${}^i\mathcal{L}_2^*$ to be parallel to y_m-axis at a safe distance from either the rear boundary of the vehicle.

C. Constrained sensor features

The set of constrained sensor features (Fig. 3c) used for collision avoidance $\mathbf{s}^c$ is defined as:

$$\mathbf{s}^c = [s_1^c, \ldots, s_{10}^c]^\mathsf{T} = [\mathbf{s}_3, \mathbf{s}_5, \mathbf{s}_6]^\mathsf{T} \tag{16}$$

with

$$\mathbf{s}_3 = [{}^3\mathbf{h}_2(3), {}^3\mathbf{h}_4(3), {}^3X_2, {}^3Y_2, {}^3d_{lat_2}]^\mathsf{T} \tag{17a}$$

$$\mathbf{s}_5 = {}^5\mathbf{h}_3(3) \tag{17b}$$

$$\mathbf{s}_6 = [{}^6\mathbf{h}_2(3), {}^6\mathbf{h}_3(3), {}^6X_3, {}^6Y_3]^\mathsf{T} \tag{17c}$$

where the difference of raddi ${}^id_{lat_a}$ is defined as:

$${}^id_{lat_a} = {}^i\rho_{p_a} - \rho_{lat}, \tag{18}$$

with:

$${}^i\rho_{p_a} = \sqrt{({}^iX_a + x_i)^2 + ({}^iY_a + y_i - \rho_m)^2}, \tag{19}$$

$$\rho_{lat} = |\rho_m| - \frac{w_{ve}}{2}. \tag{20}$$

The interaction matrices $\check{\mathbf{L}}_{i_{X_a}}$ and $\check{\mathbf{L}}_{i_{Y_a}}$ associated, respectively, to ${}^iX_a$ and ${}^iY_a$ are:

$$\check{\mathbf{L}}_{i_{X_a}} = \begin{bmatrix} -1 & 0 & {}^iY_a \end{bmatrix} \tag{21}$$

$$\check{\mathbf{L}}_{i_{Y_a}} = \begin{bmatrix} 0 & -1 & -{}^iX_a \end{bmatrix} \tag{22}$$

while interaction matrix associated to ${}^id_{lat_a}$ is defined as:

$$\check{\mathbf{L}}_{i_d} = \begin{bmatrix} 0 & \frac{{}^i\varrho_y}{{}^i\rho_{p_a}^2} & \frac{{}^iX_a\,{}^i\varrho_y}{{}^i\rho_{p_a}^2} \end{bmatrix} \tag{23}$$

with ${}^i\varrho_y = -|{}^iY_a + y_i - \rho_m|$. The interaction matrices associated to the rest of the features used as constraints can be deduced from the third row of (13).

The corresponding interaction matrix $\check{\mathbf{L}}_{\mathbf{s}}^c$ is computed at each iteration.

It should be noted that some constraints must be deactivated under certain conditions in order to be able to park successfully. For instance, the constraints on ${}^3X_2$ and ${}^6X_3$ are used to avoid collision, respectively, with points ${}^3p_2$ and ${}^6p_3$, but they would prevent the vehicle from entering the parking spot if they remain active all the time. Thus, if the vehicle is in a configuration where it can safely enter the parking spot without colliding with the aforementioned points, the previously mentioned constraints should be deactivated. Some other constraints must be deactivated under certain circumstances in order to ensure a successful, collision-free parking maneuver. The equations detailing the deactivation conditions (relying only on the sensor features and control signals) used to obtain the results presented in this work can be found in the appendix.

IV. CONTROL

The control input of the robotized vehicle is defined as:

$$\mathbf{v}_r = [v, \phi]^\mathsf{T} \tag{24}$$

with ϕ, considering (1) and (2), being mapped to $\dot{\theta}$ by

$$\dot{\theta} = \frac{v}{\rho_m}. \tag{25}$$

The MSBPC approach being explored is based on the Visual Predictive Control (VPC) described in [17].

A. Structure

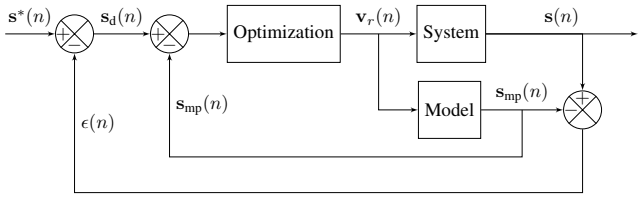


Fig. 4. Control structure [17]

The control structure is based on the internal-model-control (IMC) structure [18] (Fig. 4). The system block contains the robotized vehicle system and sensors whose input is the control variable $\mathbf{v}_r$ and output $\mathbf{s}$ is the current value of the sensor features. The reference $\mathbf{s}^*$ is the desired value of the task sensor features. The error signal ϵ represents all the modeling errors and disturbances between the current features and the values that were predicted from the model:

$$\epsilon(n) = \mathbf{s}(n) - \mathbf{s}_{\text{mp}}(n) \tag{26}$$

where n is the current time.

The optimization algorithm minimizes the difference between the desired value $\mathbf{s}_{\text{d}}$ and the predicted model output $\mathbf{s}_{\text{mp}}$. According to Fig. 4:

$$\mathbf{s}_{\text{d}}(n) = \mathbf{s}^*(n) - \epsilon(n) = \mathbf{s}^*(n) - (\mathbf{s}(n) - \mathbf{s}_{\text{mp}}(n)) \tag{27}$$

from where it is possible to deduce

$$\mathbf{s}_{\text{d}}(n) - \mathbf{s}_{\text{mp}}(n) = \mathbf{s}^*(n) - \mathbf{s}(n) \tag{28}$$

Therefore, to track $\mathbf{s}^*$ by $\mathbf{s}$ is equivalent to track $\mathbf{s}_{\text{d}}$ by $\mathbf{s}_{\text{mp}}$.

To predict the behavior of $\mathbf{s}_{\text{mp}}$ over a finite prediction horizon N_p, the interaction model described in Sec. III is

used. The difference between $\mathbf{s}_{\text{d}}$ and $\mathbf{s}_{\text{mp}}$ is used to define a cost function J to be minimized with respect to a control sequence $\tilde{\mathbf{v}}_r$ over N_p. It should be noted that only the first component $\mathbf{v}_r(n)$ of the optimal control sequence is actually applied to the vehicle.

B. Constraint handling

Model-predictive-control strategies are capable of explicitly take into into account constraints in the control-law design.

The longitudinal velocity v and steering angle ϕ are bounded by its maximum values as follows:

$$|v| < v_{\text{max}} \tag{29a}$$

$$|\phi| < \phi_{\text{max}} \tag{29b}$$

where v_{max} is an adaptive saturation value imposing a deceleration profile based on the velocity profile shown in [4] as the vehicle approaches the final pose. Furthermore, to avoid large changes in the control signals at the current iteration n that may cause uncomfortable sensations for the passengers or surrounding witnesses and, to consider to some extent the dynamic limitations of the vehicle, the control signals are saturated as well by some increments with respect to the previous control signals (at iteration $n-1$) as shown below:

$$(v_{n-1} - \Delta_{dec}) \leq v_n \leq (v_{n-1} + \Delta_{acc}) \tag{30a}$$

$$(\phi_{n-1} - \Delta_{\phi}) \leq \phi_n \leq (\phi_{n-1} + \Delta_{\phi}). \tag{30b}$$

$$(\dot{\phi}_{n-1} - \Delta_{\dot{\phi}}) \leq \dot{\phi}_n \leq (\dot{\phi}_{n-1} + \Delta_{\dot{\phi}}). \tag{30c}$$

The sensor features considered for collision avoidance (16) are constrained as follows:

$$\mathbf{s}^c_{\text{min}} \leq \mathbf{s}^c \leq \mathbf{s}^c_{\text{max}} \tag{31}$$

By writing the constraints (30) and (31) as nonlinear functions:

$$C(\mathbf{v}_r) \leq 0 \tag{32}$$

a constraint domain $\mathbb{C}$ can be defined.

C. Mathematical formulation

The MSBPC approach can be written in discrete time as follows:

$$\begin{aligned} &\min\ J(\mathbf{v}_r) \\ &\tilde{\mathbf{v}}_r \in \mathbb{C} \end{aligned} \tag{33}$$

with

$$J(\mathbf{v}_r) = \sum_{j=n+1}^{n+N_p} [\mathbf{s}_{\text{d}} - \mathbf{s}^t_{\text{mp}}(j)]^T \mathbf{Q}(j) [\mathbf{s}_{\text{d}} - \mathbf{s}^t_{\text{mp}}(j)] \tag{34}$$

and

$$\tilde{\mathbf{v}}_r = \{\mathbf{v}_r(n), \mathbf{v}_r(n+1), \ldots, \mathbf{v}_r(n+N_c), \ldots, \mathbf{v}_r(n+N_p-1)\} \tag{35}$$

subject to

$$\mathbf{s}^t_{\text{mp}}(j) = \mathbf{s}^t_{\text{mp}}(j-1) + \mathbf{L}^t_{\mathbf{s}}(j-1) T_s \mathbf{v}_m(j-1) \tag{36a}$$

$$\mathbf{s}^c_{\text{mp}}(j) = \mathbf{s}^c_{\text{mp}}(j-1) + \mathbf{L}^c_{\mathbf{s}}(j-1) T_s \mathbf{v}_m(j-1) \tag{36b}$$

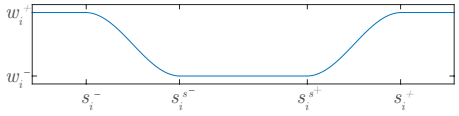


Fig. 5. Weighting function w_i^t

The weighted matrix $\mathbf{Q}$ remains constant along the prediction horizon and, in order to automatically adapt the influence of each task feature, is defined as:

$$\mathbf{Q} = \left[\begin{array}{c|c} Q_1 \operatorname{diag}(w_1^t, \ldots, w_3^t) & 0_{3\times 6} \\ \hline 0_{6\times 3} & Q_2 \operatorname{diag}(w_4^t, \ldots, w_9^t) \end{array}\right] \tag{37}$$

where w_1^t-w_3^t, w_6^t and w_9^t are constant while the values of $w_i^t\ \forall i = \{4,5,7,8\}$ and Q_2 are computed using a smooth weighting function (Fig. 5) based on the one presented in [19], while:

$$Q_1 = \begin{cases} 0 & \text{if } \|\mathbf{s}_{2_{\mathcal{L}_1}} - \mathbf{s}^*_{2_{\mathcal{L}_1}}\| < \epsilon_{\mathcal{L}_1} \\ 1 - Q_2 & \text{otherwise} \end{cases} \tag{38}$$

where $\epsilon_{\mathcal{L}_1}$ is a small positive scalar value.

Since in the parking scenarios considered, the error $\mathbf{e}^t_1 = \mathbf{s}^t_1 - \mathbf{s}^{t^*}_1$ would be generally minimized with a forward motion (particularly when the vehicle is close to the boundaries of the parking spot) while $\mathbf{e}^t_2 = \mathbf{s}^t_2 - \mathbf{s}^{t^*}_2$ with a backward one, by regulating the influence of each set of sensor features (by means of Q_1 and Q_2, respectively) the controller can automatically maneuver the vehicle with the appropriate direction of motion that would allow to have a successful parking maneuver. Regarding the use of $\epsilon_{\mathcal{L}_1}$, it serves to nullify Q_1 (and consequently the influence of $\mathbf{s}^t_1$) when the vehicle is close to be collinear to $\mathcal{L}_1$.

It should be noted that, from $\mathbf{v}_r(n+N_c)$ to $\mathbf{v}_r(n+N_p-1)$, the control input is constant and is equal to $\mathbf{v}_r(n+N_c)$, where N_c is the control horizon.

V. RESULTS

For the results shown in this section, the parameters in Table II are considered. The value of ϕ_{max} corresponds to the maximum steering angle of the real vehicle while the rest of the parameters were determined by empirical testing, nevertheless some guidelines on how to tune them can be given:

- The maximum longitudinal velocity v_{max} and the increments Δ_v, Δ_ϕ and $\Delta_{\dot{\phi}}$ should be large enough so that the vehicle can park in a reasonable amount of time (without a feeling of *sluggishness*) but not so large that the passengers and surrounding witnesses feel unease during the maneuver.
- A larger control horizon N_c allows the system to maneuver the vehicle more freely at the expense of a larger computation effort.
- N_p should be large enough so that a collision-free motion can be guaranteed (i.e. $N_p \geq v_{\text{max}}/\Delta_v$) but small enough to be able to meet the computational time requirements.
- The threshold value $\epsilon_{\mathcal{L}_1}$ used to determine whether or not Q_1 should be equal to zero has influence on the

total number of maneuvers required to park and on the convergence of the controller. In general, a smaller value of $\epsilon_{\mathcal{L}_1}$ enforces a smaller final error at the expense of an increase on the number of maneuvers required to park.

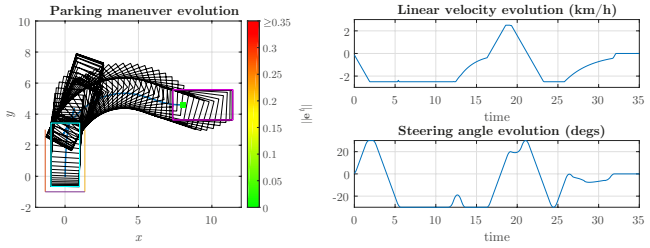
The nonlinear solver used for MATLAB implementations is fmincon with a Sequential Quadratic Programming (SQP) algorithm while for C++ implementations the solver NLopt with a Sequential Least Squares Programming (SLSQP) algorithm is used.

TABLE II

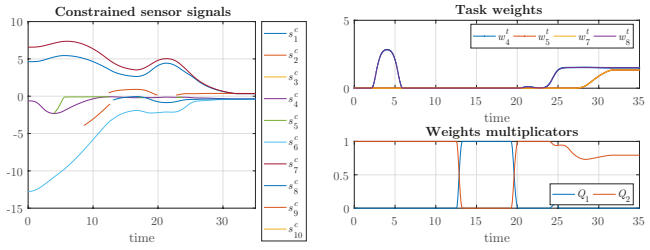
CONTROL-RELATED VEHICLE PARAMETERS

Parameteres	*Notation*	*Value*
Control horizon	N_c	4
Prediction horizon	N_p	20
Sampling time	T_s	0.1s
Maximum steering angle	$\phi_{\max}$	30°
Maximum longitudinal velocity	$v_{\max}$	$\leq$ 0.6944m/s
Maximum velocity increment	Δ_v	0.35m/s T_s
Maximum ϕ increment	Δ_ϕ	2° T_s
Maximum $\dot{\phi}$ increment	$\Delta_{\dot{\phi}}$	0.8 T_s
Threshold value to nullify Q_1	$\epsilon_{\mathcal{L}_1}$	0.125

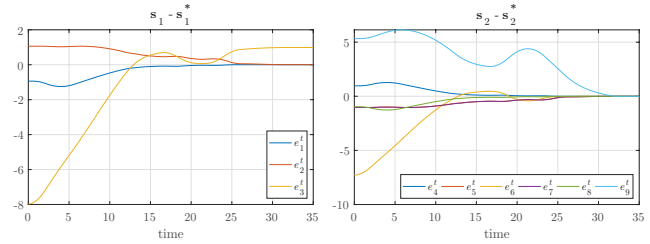
A. MATLAB simulations



(a) Performed maneuver (b) Control signals



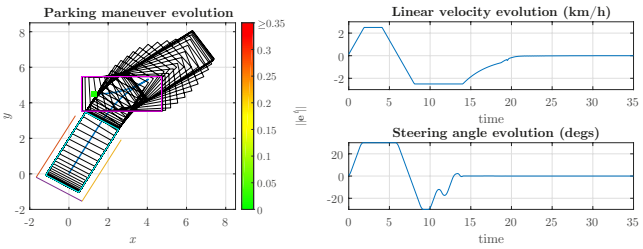
(c) Constrained sensor signals (d) Variable weighting signals



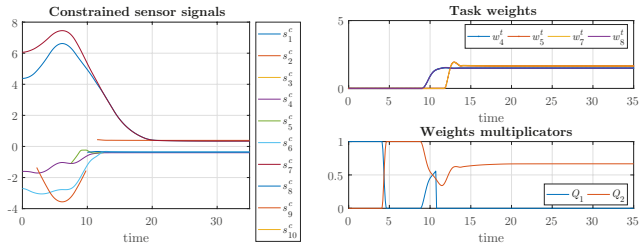
(e) Task error signal: $\mathbf{e}_1^t$ (f) Task error signal: $\mathbf{e}_2^t$

Fig. 6. Backward perpendicular parking maneuver. Initial pose = (8m, 4.6m, 0°)

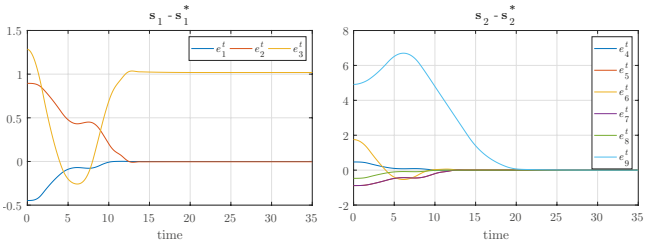
To illustrate the behavior of the MSBPC approach, a perpendicular (Fig. 6) and a diagonal (Fig. 7) maneuvers



(a) Performed maneuver (b) Control signals



(c) Constrained sensor signals (d) Variable weighting signals



(e) Task error signal: $\mathbf{e}_1^t$ (f) Task error signal: $\mathbf{e}_2^t$

Fig. 7. Backward diagonal parking maneuver. Initial pose = (1.3m, 4.5m, 0°)

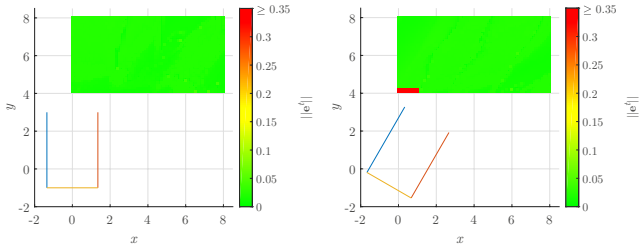
are shown. It can be clearly seen that, for both cases, the car is able to park successfully with generally smooth control signals (thanks to (30)) while satisfying the constraints on the sensor features at each time instant. Furthermore, it can be seen how, generally, when Q_2 is larger than Q_1, the vehicle is moving backward and when Q_1 is larger a transition towards a forward motion occurs, allowing the vehicle to perform multiple maneuvers in order to park successfully.

B. Exhaustive simulations

To assess the stability and convergence of the presented approach, various convergence analyses for the different parking cases were conducted by means of exhaustive simulations. Due to paper length constraints, for the two shown cases (Figs. 8a-8b), the initial orientation of the vehicle is 0°.

Since the exhaustive simulations are an aggregation of the results obtained from several simulations (like those shown in Figs. 6a and 7a), each figure consists of a parking spot (represented by 3 lines) adapted to each case and a scatter plot of the initial position of the vehicle (with a sampling step of 10cm), whose color depends on the final value of $||\mathbf{e}^t||$. The green portion of each scatter plot corresponds to the region of attraction (ROA) and the red one represents the initial positions that are outside of the ROA.

It can be clearly seen that, thanks to the capability of the MSBPC approach of performing automatically multiple



(a) Backward perpendicular case. (b) Backward diagonal case.

Fig. 8. Exhaustive simulations. Initial orientation = 0°. Parking spot length = 4m and width = 2.7m

maneuvers, the car is able to park from almost any initial position in the analysis window with the exception of a small portion on the diagonal case (Fig. 7a) where the vehicle is already violating the constraints from the initial position.

C. Fast prototyping environment

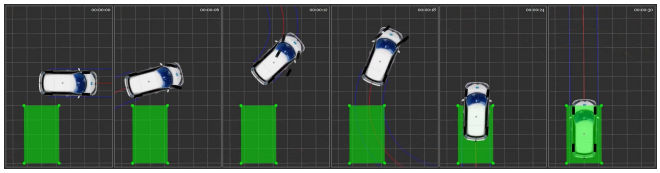
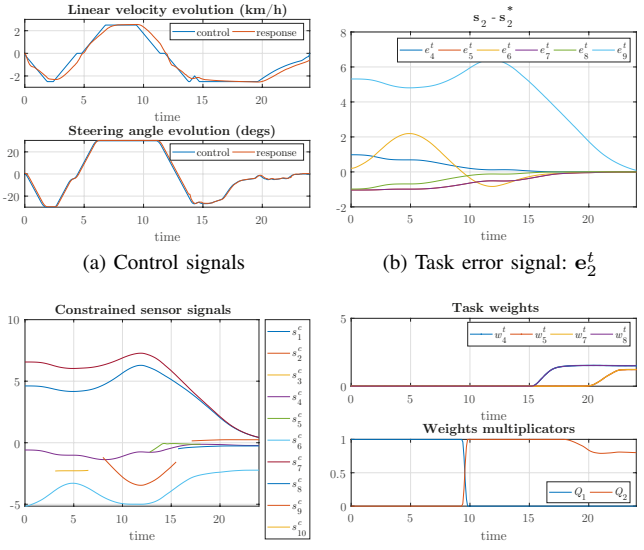


Fig. 9. Backward perpendicular parking maneuver in simulation using a homemade fast prototyping environment



(a) Control signals (b) Task error signal: $\mathbf{e}_2^t$

(c) Constrained sensor signals (d) Variable weighting signals

Fig. 10. Backward perpendicular parking maneuver signals

A homemade fast prototyping environment using the same software architecture as the one embedded inside the car is used for simulation purposes. In addition to behaving nearly identically (from a software architecture point of view) to the real vehicle, this fast prototyping environment simulates as well the dynamics of the vehicle, leading to more realistic simulations than the MATLAB environment used for the results presented in the previous subsections.

As it can be seen in Figs. 9-10, the car is able to park successfully into the parking spot (represented by a green rectangle) in three motions while satisfying the constraints during the whole maneuver, with the evolution of the many different signals being very similar to the MATLAB cases in spite of the slight discrepancy between the control signals and the response of the vehicle (Fig. 10a). The fast deceleration at the end (Fig. 10a) is due to a stopping condition in the implementation related to $\mathbf{e}_t$.

VI. CONCLUSIONS

Following our previous work [12], we've shown how the use of a prediction step makes possible to overcome the main limitation of the previously presented Multi-Sensor-Based control approach - being able to park with only one maneuver. Indeed, thanks to the prediction step considered, the presented MSBPC approach is able to successfully deal with backward perpendicular and diagonal parking problems (using the same formalism) in multiple motions from virtually any sensible initial position.

It is worth noting that the modifications in the interaction model with respect to the MSBC approach are minor.

APPENDIX

The constraints deactivation conditions used to obtain the results presented in this work are now detailed (Table III). To simplify the content of the table, the following notation is considered: subscripts $_{\text{min}}$ denotes a minimum radius when turning with the maximum steering angle (ϕ_{max}), ${}^i p_a^{\text{Cart}}$ describes the point ${}^i p_a$ in Cartesian coordinates, the superscript ${}^{\text{c(angle)}}$ denotes a multiplication of the base by $\cos(\text{angle})$ with angle expressed in degrees and, ϵ_{long} and ϵ_{lat} are small positive values considered for constraints that are mostly related to, respectively, the longitudinal or lateral motions ($\epsilon_{\text{long}} = 0.05$ and $\epsilon_{\text{lat}} = 0.1$). Furthermore, it should be noted that the conditions should be verified at each prediction step along the whole prediction horizon with the appropriate predicted value for each feature and corresponding control signal.

TABLE III
CONSTRAINTS DEACTIVATION CONDITIONS

Constraint	*Deactivate if*
${}^3\mathbf{h}_4(3)$	$!({}^3Y_2 < 0$ and ${}^6Y_3 > 0)$ or ${}^3X_2 < 0$
${}^3X_2$	${}^3x_2 < -2v_{\text{max}}^{\text{abs}}$ or ${}^3Y_2 < -\epsilon_{\text{long}}$
${}^3d_{lat_2}$	$\phi \geq 0$ or $(v < 0$ and ${}^3X_2 > -x_i)$ or $(\lvert{}^5\mathbf{h}_4(3)\rvert > \rho^{\text{c45}}_{m_{\text{min}}}$ and ${}^3p_2^{\text{Cart}} > \rho^{\text{c45}}_{m_{\text{min}}})$
${}^6\mathbf{h}_3(3)$	${}^3Y_3 < -\epsilon_{\text{lat}}$ or $({}^6Y_3 < \epsilon_{\text{lat}}$ and ${}^6X_3 < 0)$ or $({}^6X_3 > 0$ and ${}^3Y_3 < 0)$
${}^6X_3$	${}^3Y_3 < 0$ or ${}^6Y_3 > \epsilon_{\text{long}}$ or $({}^3\mathbf{h}_4(3) > 0$ and ${}^3Y_3 < 0)$
${}^6Y_3$	${}^5X_3 > 2v_{\text{max}}^{\text{abs}}$ or ${}^6X_3 < 0$ or ${}^6Y_3 > \epsilon_{\text{lat}}$

ACKNOWLEDGMENT

This work was supported by the Mexican National Council for Science and Technology (CONACYT). This paper des-

cribes work carried out in the framework of the Valet project, reference ANR-15-CE22-0013-02.

REFERENCES

[1] W. Wang, Y. Song, J. Zhang, and H. Deng, "Automatic parking of vehicles: A review of literatures," *International Journal of Automotive Technology*, vol. 15, no. 6, pp. 967–978, 2014.

[2] Y. Song and C. Liao, "Analysis and Review of State-of-the-Art Automatic Parking Assist System," in *2016 IEEE International Conference on Vehicular Electronics and Safety*, Beijing, China, 2016, pp. 61–66.

[3] P. Petrov, F. Nashashibi, and M. Marouf, "Path Planning and Steering control for an Automatic Perpendicular Parking Assist System," in *7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV'15*, Hamburg, Germany, 2015, pp. 143–148.

[4] P. Petrov and F. Nashashibi, "Saturated Feedback Control for an Automated Parallel Parking Assist System," in *13th International Conference on Control, Automation, Robotics and Vision (ICARCV'14)*, Marina Bay Sands, Singapore, 2014, pp. 577–582.

[5] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar, "Geometric Continuous-Curvature Path Planning for Automatic Parallel Parking," in *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Evry, France, 2013, pp. 418–423.

[6] Y. Yi, Z. Lu, Q. Xin, L. Jinzhou, L. Yijin, and W. Jianhang, "Smooth path planning for autonomous parking system," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, no. Iv. IEEE, 2017, pp. 167–173.

[7] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium*, Seoul, Korea, 2015, pp. 1148–1153.

[8] G. Notomista and M. Botsch, "Maneuver segmentation for autonomous parking based on ensemble learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, Ireland, 2015, pp. 1–8.

[9] D. A. de Lima and A. C. Victorino, "Sensor-Based Control with Digital Maps Association for Global Navigation: A Real Application for Autonomous Vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Las Palmas, Spain, 2015, pp. 1791–1796.

[10] Y. Kang, D. A. de Lima, and A. C. Victorino, "Dynamic obstacles avoidance based on image-based dynamic window approach for human-vehicle interaction," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Seoul, South Korea, jun 2015, pp. 77–82.

[11] D. Pérez Morales, S. Domínguez Quijada, O. Kermorgant, and P. Martinet, "Autonomous parking using a sensor based approach," in *8th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV'16 at 19th IEEE ITSC 2016*, Rio de Janeiro, Brazil, 2016, pp. 211–216.

[12] D. Pérez-Morales, O. Kermorgant, S. Domínguez-Quijada, and P. Martinet, "Laser-Based Control Law For Autonomous Parallel And Perpendicular Parking," in *Second IEEE International Conference on Robotic Computing*, Laguna Hills, CA, 2018, pp. 65–71.

[13] F. Chaumette and S. Hutchinson, "Visual servo control, part I : Basic Approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. December, pp. 82–90, 2006.

[14] N. Andreff, B. Espiau, and R. Horaud, "Visual Servoing from Lines," *International Journal of Robotics Research*, vol. 21, no. 8, pp. 679–699, 2002.

[15] B. Přibyl, P. Zemčík, and M. Čadik, "Camera Pose Estimation from Lines using Plücker Coordinates," in *Procedings of the British Machine Vision Conference 2015*. British Machine Vision Association, 2015, pp. 45.1–45.12.

[16] O. Thari and Y. Mezouar, "On the efficient second order minimization and image-based visual servoing," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, may 2008, pp. 3213–3218.

[17] G. Allibert, E. Courtial, and F. Chaumette, "Predictive Control for Constrained Image-Based Visual Servoing," *IEEE Trans. on Robotics*, vol. 26, no. 5, pp. 933–939, 2010.

[18] M. Morari and E. Zafiriou, *Robust Process Control*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[19] V. K. Narayanan, F. Pasteau, M. Marchal, A. Krupa, and M. Babel, "Vision-based adaptive assistance and haptic guidance for safe wheelchair corridor following," *Computer Vision and Image Understanding*, vol. 149, pp. 171–185, 2016.

Towards Uncertainty-Aware Path Planning for Navigation on Road Networks Using Augmented MDPs

Lorenzo Nardi Cyrill Stachniss

***Abstract*— Although most robots use probabilistic algorithms to solve state estimation problems such as localization, path planning is often performed without considering the uncertainty about robot's position. Uncertainty, however, matters in planning. In this paper, we investigate the problem of path planning considering the uncertainty in the robot's belief about the world, in its perceptions and in its action execution. We propose the use of an uncertainty-augmented Markov Decision Process to approximate the underlying Partially Observable Markov Decision Process, and we employ a localization prior to estimate how the uncertainty about robot's belief propagates through the environment. This yields to a planning approach that generates navigation policies able to make decisions according to different degrees of uncertainty while being computationally tractable. We implemented our approach and thoroughly evaluated it on different navigation problems. Our experiments suggest that we are able to compute policies that are more effective than approaches that ignore the uncertainty and also to outperform policies that always take the safest actions.**

I. Introduction

Over the past decades, there has been a great progress in autonomous robot navigation and today we find lots of robots that navigate indoors and outdoors. Although most robots use probabilistic algorithms for localization or mapping, most path planning systems assume to know the position of the robot while computing a path. Ignoring position uncertainty during planning may be acceptable if the robot is precisely localized, but it can lead to suboptimal navigation decisions if the uncertainty is large. Consider for example the belief about robot's position represented in Fig. 1 by the black shaded area (the darker the more likely). The robot could be at intersection A or B, but the localization system is not able to disambiguate them. Ignoring the uncertainty, we could assume the robot to be at the most likely position B. Thus, it should turn to the right to reach the goal through the shortest path (blue). However, if the robot is at A (less likely, but possible), going right would lead it to a detour (red).

In this paper, we investigate the problem of path planning under uncertainty. Uncertainty-aware plans reduce the risk to make wrong turns when the uncertainty is large. For example, in Fig. 1, the robot could navigate towards intersection C, which has distinctive surrounding and, thus, the robot is expected to localize better. There, it can safely turn towards the goal avoiding the risk of long detours (green). A general formalization for this type of problem is the Partially Observable Markov Decision Process (POMDP). POMDPs, however, become quickly intractable for real-world applications. Our goal is to investigate an approximation that is still able to consider the localization uncertainty.

All authors are with the University of Bonn, Germany.

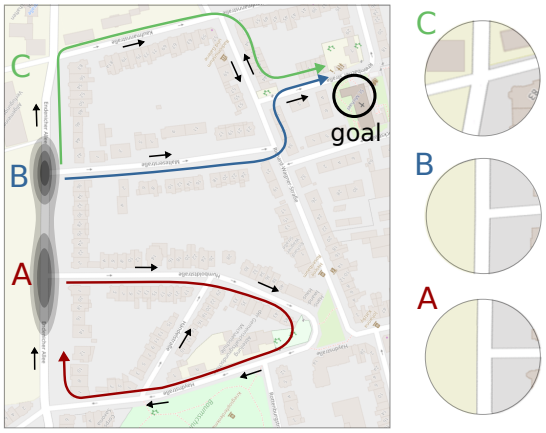


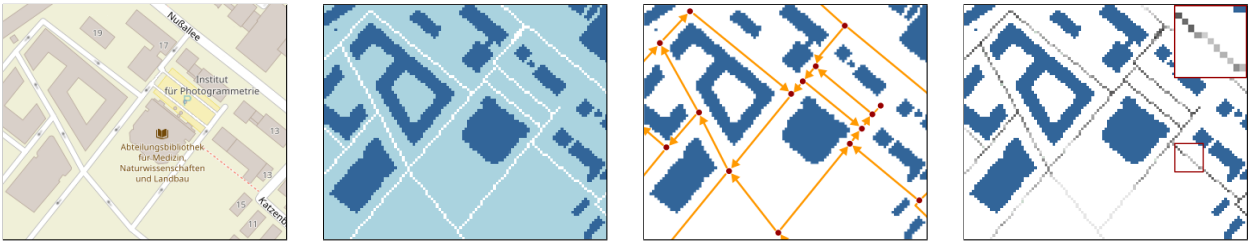
Fig. 1: Example of robot navigation with high pose uncertainty. The black shaded area is the belief about robot's position (the darker, the more likely). A, B, C are the road intersections (in detail on the right side). The paths from each intersection are colored respectively in red, orange and green. The black arrows indicate the roads' directions.

The main contribution of this paper is a novel approach that is a step forward in planning routes on road networks considering the uncertainty about robot's position and action execution. It relies on the Augmented Markov Decision Process (A-MDP) [15], which approximates a POMDP by modeling the uncertainty as part of the state. We employ a localization prior to estimate how robot's belief propagates along the road network. The resulting policy minimizes the expected travel time while reducing the mistakes that the robot makes during navigation with large position uncertainty. As a result, our planning approach, first, explicitly considers the robot's position uncertainty, and thus it is able to take different actions according to the degree of uncertainty; second, in complex situations, it leads to plans that are on average shorter than a shortest path policy operating under uncertainty but ignoring it.

II. Related Work

Although planning under uncertainty has received substantial attention, most robotic systems such as Obelix [10] or the Autonomous City Explorer [11] still use A* to navigate in urban environments. Navigation in urban environments often exploits topological or topo-metric maps [9]. These maps can be stored compactly as a graph and free maps of most cities exist, for example, through OpenStreetMap.

The Markov Decision Process (MDP) allows for optimally solving planning problems in which the actions are noisy but the state is fully observable. If the state is not fully observable, the problem turns into a Partially Observable Markov Decision Process (POMDP). However, in POMDPs,



(a) OSM map. (b) Traversability grid map $\mathcal{X}$. (c) Roads graph $\mathcal{G}$. (d) Localizability map $\mathcal{Z}$.

Fig. 2: Environment representation $\mathcal{X}, \mathcal{G}$ and localizability map $\mathcal{Z}$ extracted from OpenStreetMaps (a). In (b), the traversable roads are in white, blue refers to non-traversable areas and buildings are in dark blue. In (c), the orange arrows are the roads E and the red dots are their intersections V. In (d), the darker the pixels along the roads, the smaller the localizability covariance.

the computational complexity is often too high to provide useful results for real-world problems [12]. Roy *et al.* [15] proposed the Augmented Markov Decision Process (A-MDP) to approximate the state space of a POMDP. A-MDPs formalize POMDPs as MDPs with an augmented state representation including the uncertainty. Thus, A-MDPs can be solved using the tools of the MDP world. A-MDPs have been used by Hornung *et al.* [6] for planning while minimizing the motion blur of its camera, and by Kawano [8] to control under-actuated blimps. In this paper, we use A-MDPs to plan routes on road networks taking the uncertainty about robot's position into account.

Approaches that incorporate the robot's uncertainty into the planning process are usually referred to as planning in belief space. The belief roadmap [14] and the FIRM [1] generalize probabilistic roadmap algorithm to plan in the belief space. Platt *et al.* [13] assume maximum likelihood observations to define the belief space. The LQG-MP [2] plans using a linear-quadratic controller with Gaussian uncertainty. Most of these approaches compute a fixed path offline and execute it without considering any sensor or process noise. Our approach generates offline a policy that deals with different degrees of uncertainty, and selects online the optimal action given the current belief of the robot.

Candido *et al.* [4] and Indelman *et al.* [7] approach planning in belief space in the continuous domain. However, these approaches are computationally expensive. On the contrary, we consider a discrete space representation and use a compact representation of the robot's belief similar to Bopardikar *et al.* [3] to tackle larger environments and, thus, to take a step towards real world applications.

III. Planning and Localization in Road Networks

A. Metric-Topological Maps

Most probabilistic approaches for robot localization rely on occupancy grid maps, whereas topology graphs are an effective representation for planning. We combine these two representations and represent the environment using a metric-topological map, similar to the hierarchical maps [9].

We define our environment representation by extracting information about buildings and roads from publicly available map services such as OpenStreetMap (OSM) (see for example Fig. 2a). We store this data in a 2D grid map $\mathcal{X}$ in which each cell contains information about its traversability (Fig. 2b). We use $\mathcal{X}$ to estimate the position of the robot assuming it always moves along the roads. In addition to that, we define a topological graph $\mathcal{G} = (V, E)$ over the discretized metric space of $\mathcal{X}$ in which the vertexes $V \subset \mathcal{X}$ are the road intersections and the oriented edges E are the roads connecting them (Fig. 2c). We use $\mathcal{G}$ for planning routes. Note that an edge of $\mathcal{G}$ corresponds to the sequences of traversable cells in $\mathcal{X}$ representing the corresponding road.

B. Localization System

We consider a mobile robot equipped with a 360-degree range sensor that uses a Markov localization system [5] to localize in $\mathcal{X}$. Markov localization estimates the robot's position by considering a probability distribution over $\mathcal{X}$ in form of a histogram over all cells of the grid map, without requiring probabilities to be restricted to any particular class of distributions. As the robot moves and acquires a new scan from the laser range finder, the localization system uses the scan and the wheel odometry to estimate the new position of the robot using Bayes filter.

C. Localizability Map

Given the buildings' footprints and the sensor model of the laser range finder, we can compute an estimate of how scans fired at a location will affect the localization. We compute this prior using the method proposed by Vysotska and Stachniss [16]. It simulates at each location a virtual laser scan by ray-casting the map of the buildings. Then, it translates/rotates the virtual sensor and estimates the error between the scan and the map around its firing location. Considering these errors, it computes a covariance matrix that estimates how well the scan matches the map under position uncertainty. At locations where the surrounding environment has a distinctive structure, the resulting covariance is small, whereas it is large if the surrounding environment is not informative or ambiguous. We compute this prior for each traversable cell in $\mathcal{X}$ and we refer to this as the localizability map $\mathcal{Z}$ (see for example Fig. 2d).

D. MDP-based Planning

Given our representation of the environment $\mathcal{G}$, we can plan routes using a Markov Decision Process (MDP) in which the states are the road intersections V and the actions correspond to selecting roads E at intersections. The transition function allows for transitions between intersections

if a road connecting them, and the rewards correspond to the length of the roads. Solving this MDP generates navigation policy that leads the robot to the goal through the shortest path. However, MDPs assume to always know the location of the robot, and this is often not the case in robot navigation. Thus, following a MDP policy in situations with high position uncertainty may lead the robot to take the wrong way and thus to reach the goal through a longer path.

IV. Our Approach to Planning in Road Networks Considering Localization Uncertainty

We propose to improve decision making at intersections by integrating into the planning process the uncertainty about robot's position provided by the localization system. We formulate this planning problem using Augmented MDP (A-MDP) [15]. It efficiently approximates a POMDP by augmenting the conventional MDP state with a statistic about the uncertainty, such as its entropy or covariance. Due to the augmented state representation, transition and reward functions become more complex, but, in their final formulation, A-MDPs have an analogous representation as MDPs, except for a larger number of states. Thus, they can be solved by using the same algorithms as MDPs such as policy iteration.

A. States

Even though our localization system can potentially generate any kind of probability distribution due to its non-parametric nature, we approximate the uncertainty about robot's position *during planning* by a Gaussian distribution with isotropic covariance, and we augment the MDP states with the corresponding variance. Therefore, we define an *augmented state* s as the pair $s = (v, \sigma^2)$ that corresponds to the normal distribution $\mathcal{N}(v, \Sigma)$ defined over $\mathcal{X}$ with $\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$. This representation keeps the state space compact by augmenting it by only one dimension and, thus, avoids planning to explode in complexity. As the state s represents a distribution over the discrete space $\mathcal{X}$, we also refer to it as the probability mass function $p(x \mid \mathcal{N}(v, \Sigma))$ or, equivalently, $p(x \mid s)$.

The set of augmented states S is

$$S = \{(v, \sigma^2) \mid v \in V, \sigma^2 \in W\}, \tag{1}$$

where W is a set of variances that discretizes the possible degrees of uncertainty.

B. Actions

In our A-MDP, performing an action corresponds to take a direction at a road intersection, analogously as in MDPs. We assume that every road intersection is a junction of up to 4 roads corresponding to the four cardinal directions. Thus, the set of actions is $A = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$.

C. Transition Function

The A-MDP transition function $T(s' \mid s, a)$ takes as input an augmented state $s \in S$ and an action $a \in A$, and maps it to a probability distribution of possible A-MDP end states $s' \in S$. As our A-MDP states represent probability distributions, the transition function is more complex to define compared to standard MDPs. We define T in three steps:

1) We compute the posterior probability about robot's position given that it executes a from an intersection v, to which we refer as the *posterior from an intersection* $p(x \mid v, a)$.
2) We compute the *posterior from a state* $p(x \mid s, a)$ given that the belief about the input position of the robot is represented by the state s by combining the possible posteriors from intersections according to s.
3) We map the posterior from a state into our A-MDP state representation to define the *state transitions* $T(s' \mid s, a)$.

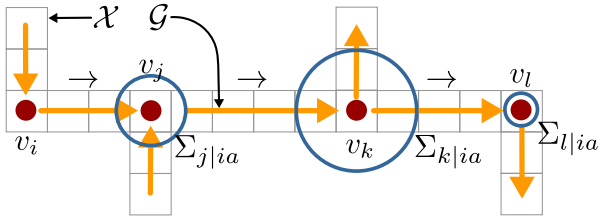


Fig. 3: Estimated uncertainty about robot's position (blue circles) at intersections v_j, v_k, v_l (red dots) when the robot takes action $a = '\!\rightarrow'$ in v_i represented in the grid map $\mathcal{X}$ underlying the road graph $\mathcal{G}$.

Posterior from an intersection: First, we compute the posterior probability about robot's position $p(x \mid v, a)$, $x \in \mathcal{X}$ given that it executes a at v without considering any uncertainty in its input position.

To this end, we *simulate* the robot taking action a at v and moving along the corresponding road in $\mathcal{X}$ according to

$$x_t = g(x_{t-1}, u_t) + \epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,\ M_t), \tag{2}$$

where g is a linearizable function, u_t is the one-step control corresponding to action a and M_t is the motion noise. Assuming that the belief about robot's position can be approximated as a Gaussian distribution, we estimate the position of the robot while navigating along a road using the prediction step of the Extended Kalman Filter (EKF)

$$p(\hat{x}_t \mid x_{t-1}, u_t) = \mathcal{N}(\hat{\mu}_t,\ \hat{\Sigma}_t) \tag{3}$$

where $\hat{\mu}_t = g(\mu_{t-1}, u_t)$, $\hat{\Sigma}_t = G_t \Sigma_{t-1} G_t^\top + M_t$, and G_t is the Jacobian of g. As we simulate robot navigation, we do not have measurements to correct the EKF prediction. Instead, we estimate how position uncertainty propagates along the road by combining $\hat{\Sigma}_t$ with the localizability covariance $\Sigma_{\hat{\mu}_t, \mathcal{Z}}$ that estimates how much informative would be a measurement at $\hat{\mu}_t$ to localize the robot:

$$p(x_t \mid x_{t-1}, u_t, \mathcal{Z}) = \mathcal{N}(\hat{\mu}_t,\ (\hat{\Sigma}_t^{-1} + \Sigma_{\hat{\mu}_t, \mathcal{Z}}^{-1})^{-1}). \tag{4}$$

If intersection v_j is reachable from v_i through an action a as in Fig. 3, we estimate the posterior probability about robot's position of executing this action as the Gaussian distribution $\mathcal{N}(v_j, \Sigma_{j|ia})$ that we compute by recursively applying Eq. (4) along the cells of $\mathcal{X}$ belonging to the corresponding road.

We explicitly model the possibility that the robot might miss an intersection and end up in a successive one while navigating with high position uncertainty. For example,

in Fig. 3, while navigating rightwards from v_i, the robot could fail to detect v_j and end up in v_k or in v_l. We compute the probability to detect the intersection v_j so that the smaller the uncertainty $\Sigma_{j|ia}$, the higher the probability to detect it:

$$p_{\text{detect}}(v_j \mid v_i, a) = p(x = v_j \mid \mathcal{N}(v_j, \Sigma_{j|ia})). \tag{5}$$

We compute the posterior $p(x \mid v_i, a)$ of taking action a at intersection v_i by considering the probability to end up in each of the reachable intersections taking action a:

$$\begin{aligned} p(x \mid v_i, a) = &\sum_{j=1}^{|J|} p(x \mid \mathcal{N}(v_j, \Sigma_{j|ia}))\, p_{\text{detect}}(v_j \mid v_i, a) \\ &\cdot \prod_{k=1}^{j-1} (1 - p_{\text{detect}}(v_k \mid v_i, a)), \end{aligned} \tag{6}$$

where J is the ordered set of $|J|$ subsequent intersections that the robot may reach by missing the previous intersections. The probability that the robot ends up in each of the J intersections decays according to the probability that a previous one has been detected. If no road exists for executing a at v, we set the posterior to be equal to the input intersection v.

Posterior from a state: Given the posteriors from the intersections, we compute the posterior probability of taking action a given that input belief about the position of the robot is the probability represented by A-MDP state $s \in S$. As the input is a probability distribution about the robot's position, the posterior from a state should represent all of the possible transitions that might occur by executing action a. Thus, we compute the posterior from a state as the weighted sum of the posteriors from the intersections according to the input state s:

$$p(x \mid s, a) = \eta \sum_{i=1}^{|V|} p(x \mid v_i, a)\, p(x = v_i \mid s). \tag{7}$$

State Transitions: We define the transition probability between the A-MDP states by computing a correspondence between the posteriors from the states and the A-MDP states S using the Bhattacharyya distance. The Bhattacharyya distance $D_B(p, q)$ measures the similarity between two distributions p and q over the same domain. We define the state transition $T(s' \mid s, a)$ with $s, s' \in S$ according to the Bhattacharyya distance over the domain $\mathcal{X}$ between the posterior $p(x \mid s, a)$ and the distribution represented by s':

$$T(s' \mid s, a) = \eta\, e^{-D_B(p(x|s,a), s')}, \tag{8}$$

where η is a normalization factor and we use the softmax function to transform the distances into probabilities.

D. Reward Function

We define the A-MDP reward function such that the resulting policy makes uncertainty-aware decisions that lead the robot to the goal in average in the minimum time or, equivalently, maximum negative time. Similarly as for the transition function, we first compute the rewards without uncertainty in the input and end position, that we call *reward between intersections* rThen, we combine the rewards between the intersections to define the A-MDP reward function R.

Assuming that the robot moves with unitary velocity, we define the reward $r(v_i, a, v_j)$ of taking action $a \in A$ from v_i to v_j with $v_i, v_j \in V$ similarly to the MDP reward:

$$r(v_i, a, v_j) = -\ell_{\text{road}}(v_i, a, v_j), \tag{9}$$

where $\ell_{\text{road}}(v_i, a, v_j)$ indicates the length of the road that connects v_i to v_j taking action a. If v_j is not reachable from v_k by taking the action a, we give a penalty as reward

$$r(v_i, a, v_k) = r_{\text{noroad}},\ r_{\text{noroad}} < 0. \tag{10}$$

For each intersection v_i that brings the robot to the goal $v_{\text{goal}} \in V$ through action a, we give a positive reward

$$r(v_i, a, v_{\text{goal}}) = r_{\text{goal}} - r(v_i, a, v_j),\ r_{\text{goal}} \geq 0. \tag{11}$$

We define the reward of taking action a from the A-MDP state s to s', with $s, s' \in S$, by combining the rewards between intersections r according to the distributions corresponding to the input and end states to reflect the uncertainty of the transitions:

$$\begin{aligned} R(s', a, s) = &\sum_{i=1}^{|V|} p(x = v_i \mid s) \\ &\cdot \sum_{j=1}^{|V|} p(x = v_j \mid s')\, r(v_i, a, v_j). \end{aligned} \tag{12}$$

E. Solving the A-MDP

In our planning problem we deal in general with non-deterministic transitions. Thus, we compute a policy that tells the robot which action to select at any intersection it might reach. As the A-MDP formulation allows for solving our planning problem as an MDP, we compute the optimal policy π^* using the policy iteration algorithm. Solving A-MDPs has the same computational complexity as MDPs but A-MDPs require a larger number of states, $|S| = |V| \cdot |W|$. POMDPs are PSPACE-complete [12], thus A-MDPs are practically and theoretically much more efficient than POMDPs.

F. Navigation Following an A-MDP Policy

At each step of the robot, the localization system computes an estimate $bel(x)$ over $\mathcal{X}$ about the robot's position as described in Sec. III-B. When the robot recognizes to be at an intersection, it has to make a decision where to navigate. In order to make decisions according to our optimal policy π^*, we transform $bel(x)$ into the A-MDP state $s \in S$ with the minimum Bhattacharyya distance:

$$s_{\text{bel}} = \underset{s \in S}{\text{argmin}}\ D_B(bel(x), s). \tag{13}$$

Thus, the robot takes the action corresponding to the optimal policy $a^* = \pi^*(s_{\text{bel}})$ and keeps navigating along the selected road until it detects the next intersection.

V. Experimental Evaluation

The objective of this work is a planning approach for robot navigation on road networks that explicitly takes the uncertainty about robot's position into account. Our experiments aim at showing that our planner makes different effective navigation decisions depending on the robot's uncertainty, the environment, and the goal location to reach. We furthermore provide comparisons to two baseline approaches.

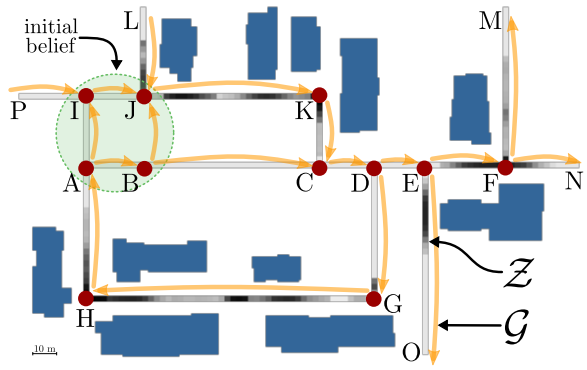


Fig. 4: Environment of Exp. 1: In the graph representation $\mathcal{G}$, intersections are the red dots denoted by letters, whereas road edges are the orange arrows. Buildings are colored in blue. Localizability information $\mathcal{Z}$ along roads is represented such that the darker, the higher the expected localization.

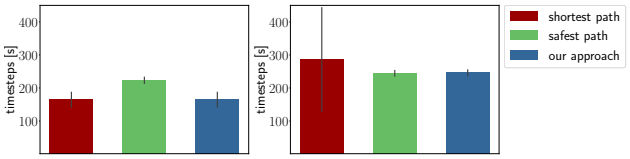


Fig. 5: Avg. travel time of Exp. 1 to the goal F (left) and G (right).

A. Simulator and Baseline

All experiments presented here are simulation experiments. The simulator uses a grid map containing buildings and road information. The robot navigates along the roads and uses building information to simulate laser range observations as well as to compute the localizability map as described in Sec. III-C. The scans and the odometry are affected by noise. The navigation decisions at the intersections are non-deterministic and the probability of missing an intersection is proportional to the variance of the robot's belief. The robot localization system implements Markov localization as described in Sec. III-B.

For comparisons, we consider a shortest path policy similar to the one described in Sec. III-D that assumes the robot to be located at the most likely position given by the localization system. We compare our approach also against a safest decision policy that considers the localizability information to reduce the expected uncertainty about robot's position and by selecting always safe actions.

B. Situation-Aware Action Selection

The first experiment (Exp. 1) is designed to show that our approach reacts appropriately to the situation given the planning problem. Fig. 4 depicts an environment together with the localizability information $\mathcal{Z}$ along the roads. According to the localizability information, the robot expects to localize well along some roads such as $\overline{\mathrm{JK}}, \overline{\mathrm{KC}}$, but finds little structure to localize in others as $\overline{\mathrm{AB}}, \overline{\mathrm{BC}}$ causing a growth in the position uncertainty. Given the initial belief that the robot is at A, B, I, or J with uniform probability (green ellipse), we sample accordingly the actual initial location, and consider two different navigation tasks to show how our approach adapts the action selection depending on the planning problem.

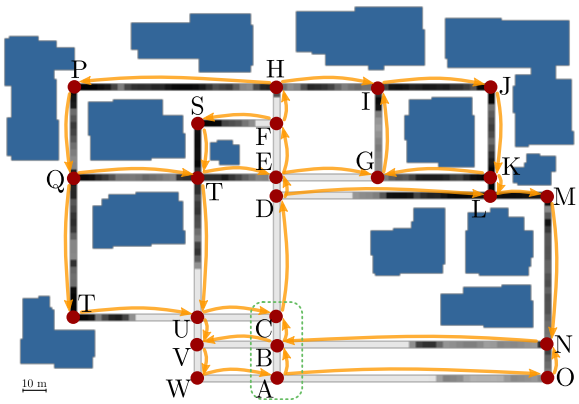


Fig. 6: Environment of Exp. 2: Same notation as in Fig. 4.

First, we set F as the goal location. The shortest path policy seeks to navigate rightwards to reach the goal fast, whereas the safest path policy seeks to go through $\overline{\mathrm{JK}}$ where the localizability is high. The policy generated by our planner performs similarly to the shortest path one. In fact, although the robot cannot localize perfectly along $\overline{\mathrm{AE}}$, it is expected to relocalize along $\overline{\mathrm{EF}}$ and thus to reach safely the goal even following a greedy plan. Fig. 5 (left) shows the average travel time of the three policies. Our policy presents the same performances as the shortest path and outperforms the safest path policy.

The situation changes if we set G as the goal and assume a time penalty corresponding to a long detour if the robot navigates towards O or N. The safest path policy seeks again to go through $\overline{\mathrm{JK}}$ to reduce the uncertainty and take the correct turn at D. Whereas, the shortest path policy leads the robot rightwards to quickly reach D and make the turn to the goal. However, navigating along $\overline{\mathrm{AD}}$, the uncertainty about robot's position grows and, thus, it increases the probability that the robot takes the wrong turn or misses the intersection D. This leads to an overall suboptimal performance, see Fig. 5 (right). As reaching D with large uncertainty may lead the robot to long detours, our planner seeks to reduce the uncertainty before making the turn and, thus, in this case, behaves similarly to the safest path policy. This shows that our planner adapts to the situation by picking the best of both the shortest and the safest path worlds.

C. Uncertainty-Aware Action Selection

The second experiment (Exp. 2) is designed to illustrate how our approach deals with different degrees of uncertainty. To do so, we consider the environment depicted in Fig. 6. The robot starts from A, B, and C with different initial levels of position uncertainty and navigates to the goal G.

Trivially, the shortest path to the goal is to navigate upwards and make a right turn to the goal at E. When the robot is accurately localized, following this path leads it fast and safely to the goal. However, as there is little structure to localize in the environment along $\overline{\mathrm{AE}}$, the uncertainty about the robot's position upon reaching E grows. Reaching E with large uncertainty increases the probability to mismatch the intersections D and E. If the robot expects to be at E

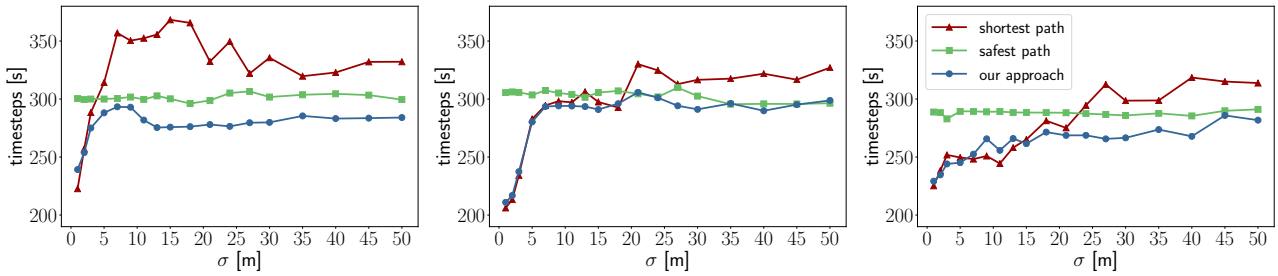


Fig. 7: Avg. travel time of Exp. 2 (Fig. 6) to reach the goal G starting from A, B, C respectively with different uncertainty levels σ.

whereas it is actually at D, the shortest path policy makes the robot turn right leading it to a long detour through L. Large uncertainty increases also the probability that the robot misses to detect E or even F leading also to detours.

The safest path policy seeks to make safe turns at intersections in which the robot is expected to localize well, for example, at the end of the roads or where the localizability is high. Therefore, to reach the goal, it leads the robot upwards to H and makes a safe right turn towards I. From I, it moves the robot rightwards to J, turns to K and, finally, to the goal G. However, the safest path policy always makes safe decisions ignoring the uncertainty about the robot's position while executing the plan. Therefore, it leads the robot through a conservative (and often longer) path also in the situations in which the position uncertainty is small.

Our approach makes decisions by explicitly considering the uncertainty about the position of the robot provided by the localization system. Thus, depending on the degree of uncertainty, it selects the action that leads the robot to the goal trading off safety and travel time.

Fig. 7 shows the performance of the three algorithms in Exp. 2. We considered 18 different levels of uncertainty with σ ranging from 1 to 50 meters and performed for each initial location and uncertainty 200 runs. The safest path policy presents in average similar travel time when varying the initial uncertainty. The shortest path policy shows short travel time when the uncertainty is small but, when the uncertainty grows, it takes in average longer than the safest path to reach the goal. Our approach follows a strategy similar to the shortest path when the uncertainty is small and thus mistakes are unlikely. However, in tricky situations when the uncertainty becomes large, our approach makes decisions similarly to the safest path, thereby avoiding long detours. Therefore, our approach is able to take the appropriate navigation action according to the degree of uncertainty, overall outperforming the shortest and safest path policies.

VI. Conclusion

In this paper, we presented a step towards efficient path planning under uncertainty on road networks. We formulate this problem as an augmented Markov Decision Process that incorporates the robot's position uncertainty into the state space but does not require solving a full POMDP. We define the transition function of the A-MDP by estimating how the robot's belief propagates along the road network through the use of a localization prior. During navigation, we transform the belief provided by the robot's localization system into our state representation to select the optimal action. Our experiments illustrate that our approach performs similarly to the shortest path policy if the uncertainty is small, but outperforms it when the uncertainty is large and the risk of making suboptimal decisions grows. Therefore, our approach is able to trade off safety and travel time by exploiting the knowledge about the robot's uncertainty.

References

[1] A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato. Firm: Feedback controller-based information-state roadmap-a framework for motion planning under uncertainty. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[2] J. Van Den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *Intl. Journal of Robotics Research*, 30(7):895–913, 2011.

[3] S. Bopardikar, B. Englot, and A. Speranzon. Robust belief roadmap: Planning under uncertain and intermittent sensing. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[4] S. Candido and S. Hutchinson. Minimum uncertainty robot navigation using information-guided pomdp planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[5] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Journal on Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

[6] A. Hornung, H. Strasdat, M. Bennewitz, and W. Burgard. Learning efficient policies for vision-based navigation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[7] V. Indelman, L. Carlone, and F. Dellaert. Planning under uncertainty in the continuous domain: a generalized belief space approach. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[8] H. Kawano. Study of path planning method for under-actuated blimp-type uav in stochastic wind disturbance via augmented-mdp. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2011.

[9] K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[10] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous Robot Navigation in Highly Populated Pedestrian Zones. *Journal of Field Robotics (JFR)*, 2014.

[11] G. Lidoris, F. Rohrmuller, D. Wollherr, and M. Buss. The autonomous city explorer (ace) project – mobile robot navigation in highly populated urban environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[12] C. Papadimitriou and J. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[13] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[14] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Intl. Journal of Robotics Research (IJRR)*, 28(11-12):1448–1465, 2009.

[15] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[16] O. Vysotska and C. Stachniss. Improving SLAM by Exploiting Building Information from Publicly Available Maps and Localization Priors. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, 85(1):53–65, 2017.

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Round table

Autonomous shuttles and taxis

- **Robert Krutsch (Intel Deutschland GmbH, Germany)**
- **Moritz Werling (BMW, Germany)**
- **Nicole Camous (NavyaTech, France)**
- **Dominik Maucher (Bosch, Germany)**

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems