2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

IROS12 4th International workshop on

Planning, Perception and Navigation for Intelligent Vehicles

Full Day Workshop

October 7th, 2012 Vilamoura, Algarve, Portugal

http://ppniv12.irccyn.ec-nantes.fr/

Organizers

Pr Alberto Broggi (VISLAB, Italy),
Pr Christian Laugier (INRIA, France),
Pr Philippe Martinet (IRCCYN, France),
Pr Urbano Nunes (ISR, Portugal)
Pr Christoph Stiller (KIT, Germany)

Contact

Professor Philippe Martinet
Ecole Centrale de Nantes,
IRCCYN-CNRS Laboratory
1 rue de la Noë, BP 92101, 44321 Nantes Cedex - FRANCE
Phone: +33 240 376 975, Sec : +33 240 376 900, Fax : +33 240 376 6934
Email: Philippe.Martinet@irccyn.ec-nantes.fr
Home page: http://www.irccyn.ec-nantes.fr/~martinet

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Foreword

Autonomous driving and navigation is a major research issue which would affect our lives in near future. 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. These new technologies can be applied efficiently for other application field such as unmanned vehicles, mobile service robots, or mobile devices for motion assistance to elderly or disable peoples. Technologies related to this area, such as autonomous outdoor vehicles, achievements, challenges and open questions would be presented, including the following topics: 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, Driver assistance systems, Environment perception, vehicle localization and autonomous navigation, Real-time perception and sensor fusion, SLAM in dynamic environments, Real-time motion planning in dynamic environments, 3D Modelling and reconstruction, Human-Robot Interaction, Behavior modeling and learning, Robust sensor-based 3D reconstruction, Modeling and Control of mobile robot, Multi-agent based architectures, Cooperative unmanned vehicles (not restricted to ground transportation), Multi autonomous vehicles studies, models, techniques and simulations.

Previously, seven 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), and the third edition PPNIV'09 was in Saint-Louis (around 70 attendees) during IROS'09 . In parallel, we have also organized SNODE'07 in San Diego during IROS'07 (around 80 attendees), SNODE'09 in Kobe during ICRA'09 (around 70 attendees), and RITS'10 in Anchrorage during ICRA'10 (around 35 attendees), and the last one PNAVHE11 in San Francisco during the last IROS11(around 50 attendees).

This workshop is composed with 4 invited talks and 15 selected papers (8 selected for oral presentation and 7 selected for interactive session. Five sessions have been organized:

- Session I: Localization & mapping
- Session II: Multiple Vehicles/Robots & Interaction
- Session III: Interactive session
- Session IV: Navigation, Control, Planning
- Session V: Perception & Situation awareness

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/).

Alberto Broggi, Christian Laugier, Philippe Martinet, Urbano Nunes and Christoph stiller

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session I

Localization & mapping

- **Keynote speaker: Philippe Bonnifait (UTC, Compiegne, France)**
  Title: Navigable Maps for Intelligent Vehicles Localization and Perception

- **Title: Robot Localization using efficient planar features matching**
  Authors: B. Charmette, E. Royer, Frédéric Chausse and L. Lequievre

- **Title: Application of Visual-Inertial SLAM for 3D Mapping of Underground Environments**
  Authors: A. Ferreira, J. Almeida and E. Silva

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session I

Keynote speaker: **Philippe Bonnifait**
(UTC, Compiegne, France)

Navigable Maps for Intelligent Vehicles Localization and Perception

Abstract : Intelligent Vehicles are robotic systems that assist the driver in safe and comfortable operation by providing pertinent information or by controlling the vehicle itself. Real-time and safe perception of the driving environment is one of the key issues. Recent evolutions of navigable maps make them suitable to assist localization and perception processes since they provide additional information that can be exploited with anticipation. This talk focuses on some autonomous techniques that merge the map information with on-board sensors data like GPS measurements, CAN -bus proprioceptive sensors, exteroceptive cameras and multi-layers lidars. Macro-scale maps with poly-lines representation of the road network can be exploited as an a priori knowledge in order to enhance GPS availability, particularly in urban canyons where satellites signals are often blocked. Such kind of map technology is also planned to be used for Map-aided ADAS (Advanced Driver Assistance Systems). However, maps can be obsolete or contain errors, resulting in malfunctions of context-based ADAS and possibly generating hazardous situations. The talk will present a sequential fault detection test able to detect and localise map errors in an autonomous manner using the on-board sensors. Meso-scale maps provide more refined information that describes the drivable space of the roads. The talk will present how 3-D facets geometry can be used for contracting East, North and altitude estimates when solving a localization problem. The use of this kind of 3D representation to characterize the drivable space (useful for path planning or obstacle avoidance) will be also presented and discussed. Finally, the talk will focus on visual landmarks that can be managed in a specific layer of the map. A method for mobile mapping lane markings and exploiting them in dynamic localization will be described. Experimental results showing the key role of navigable maps for intelligent vehicles localization and perception will be systematically presented.

Biography: Philippe Bonnifait joined the University of Technology of Compiegne (UTC) 1998. He is now a Professor in the Computer Science and Engineering Department. Prof. Bonnifait received the Electrical Engineering degree (French Engineer degree) from the Ecole Supérieure d'Electronique de l'Ouest (ESEO - Angers- France) in 1992. He graduated from the Ecole Centrale de Nantes (ECN) in 1994 (Master of Science degree). He received the Ph. D. degree in Automatic Control and Computer Science from the ECN in 1997. In December 2005, he obtained the Habilitation à Diriger des Recherches from the University de Technology of Compiegne (UTC). He joined the Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN UMR 6597), France, in 1993. In 1998, he was recipient of the Novatlante Award from the district of Nantes for his Ph.D on Outdoor Mobile Robots Localization using Goniometry. Between 1998 and 2007, Ph. Bonnifait has been Maitre de Conferences in the Computer Science and Engineering Department of UTC. He is currently full Professor. He is also with the Heudiasyc Laboratory, a joint research unit between CNRS and UTC. He is head of the research group "Automation, Embedded Systems and Robotics" (ASER) of the lab Heudiasyc. He is an IEEE and French SIA Member.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Navigable Maps for Intelligent Vehicles Localization and Perception

Philippe Bonnifait

Heudiasyc UMR7253 CNRS
Université de Technologie de Compiègne

France

Abstract

Intelligent Vehicles assist the driver in safe and comfortable operation by providing pertinent information or by controlling the vehicle itself. Real-time and safe perception of the driving environment is one of the key issues. Recent evolutions of navigable maps make them suitable to assist localization and perception processes since they provide additional information that can be exploited with anticipation.

This talk focuses on some autonomous techniques that merge the map information with on-board sensors data like GPS measurements, CAN –bus proprioceptive sensors, exteroceptive cameras and multi-layers lidars.

Macro-scale maps with poly-lines representation of the road network can be exploited as an a priori knowledge in order to enhance GPS availability, particularly in urban canyons where satellites signals are often blocked. Such kind of map technology is also planned to be used for Map-aided ADAS (Advanced Driver Assistance Systems). However, maps can be obsolete or contain errors, resulting in malfunctions of context-based ADAS and possibly generating hazardous situations. The talk will present a sequential fault detection test able to detect and localise map errors in an autonomous manner using the on-board sensors.

Meso-scale maps provide more refined information that describes the drivable space of the roads. The talk will present how 3-D facets geometry can be used for contracting East, North and altitude estimates when solving a localization problem. The use of this kind of 3D representation to characterize the drivable space (useful for path planning or obstacle avoidance) will be also presented and discussed.

Finally, the talk will focus on visual landmarks that can be managed in a specific layer of the map. A method for mobile mapping lane markings and exploiting them in dynamic localization will be described.

Experimental results showing the key role of navigable maps for intelligent vehicles localization and perception will be systematically presented.

Outline

Localization Quality of Service

Map Technology

Tightly coupled GNSS/map localization

Error detection and localization

Enhancing maps with lane marking

Experimental results

Conclusion

Localization Quality of Service

International Civil Aviation Organization

4 attributes

- Accuracy
- Integrity
- Availability
- Continuity of service

Accuracy

The degree of conformity of the position provided by the navigation system relative to the actual value.

Integrity

A measure of the trust that can be put in the information from the navigation system, i.e., the likelihood of undetected failures in the specified accuracy of the system.

Localization uncertainty

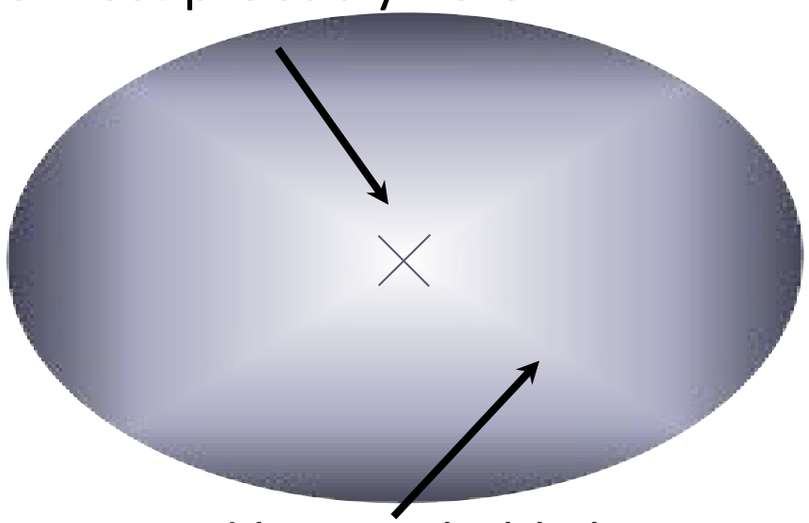
Knowledge of positioning uncertainty is necessary to decide if position information is relevant for the current application

Localization uncertainty is time and location dependent

Geometry - Satellites - Noise - Faults

Second order moments

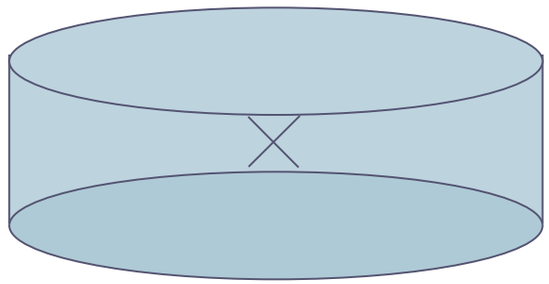
You are most probably here...



...and less probably here

Protection levels

You are somewhere inside this box

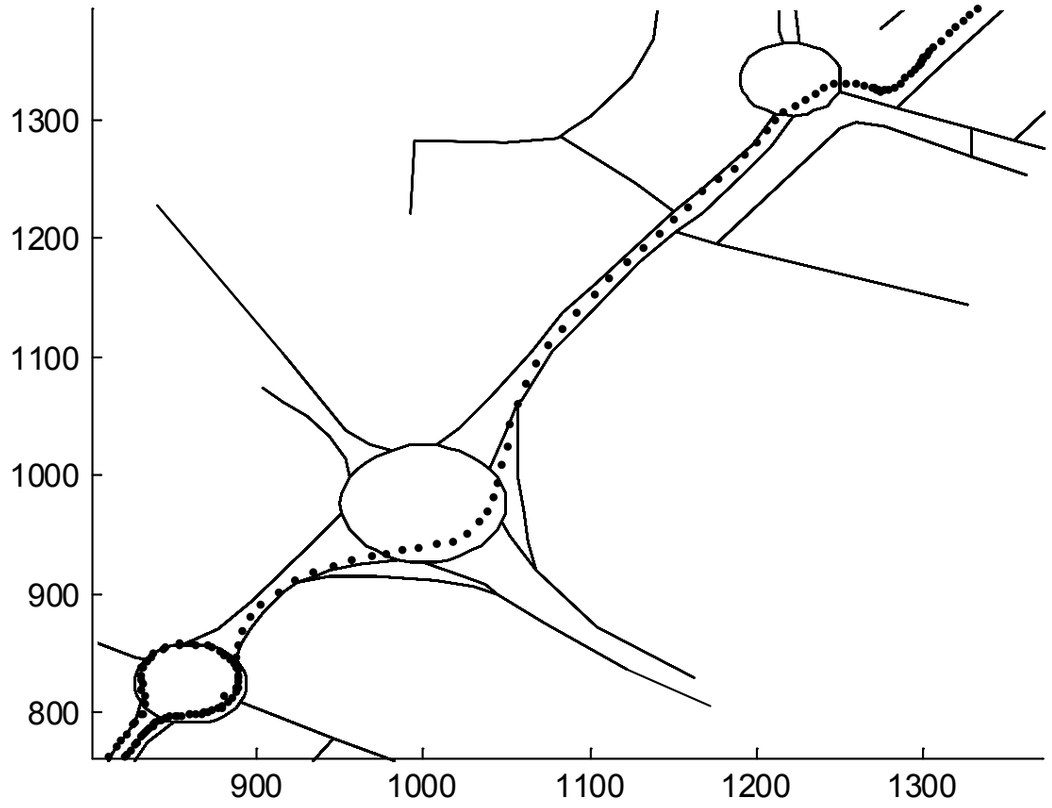


Map Technology

For road network representation

Usual representation

- Poly-lines with
  - nodes (connectivity)
  - shape points (geometry)
- Often 2D geographic coordinates (WGS84)

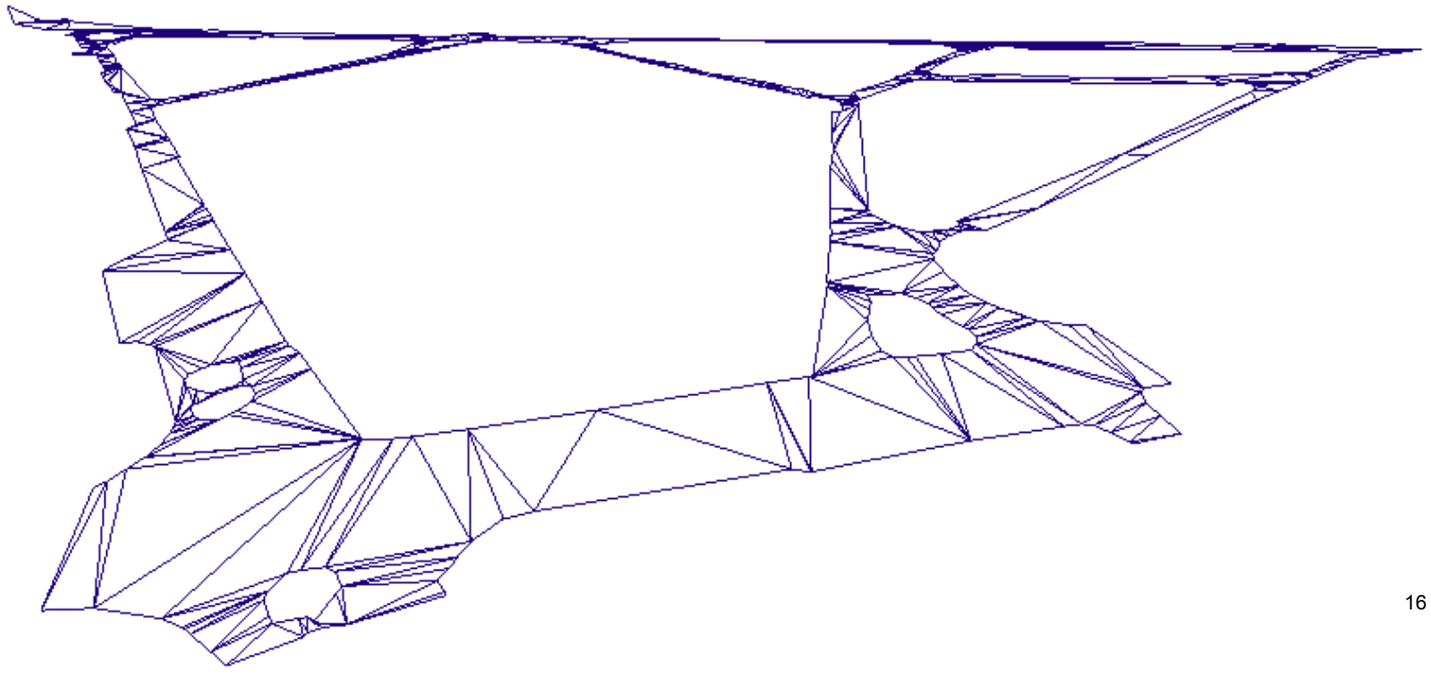


3D map of the drivable space

Produced by the French *Institut Géographique National*

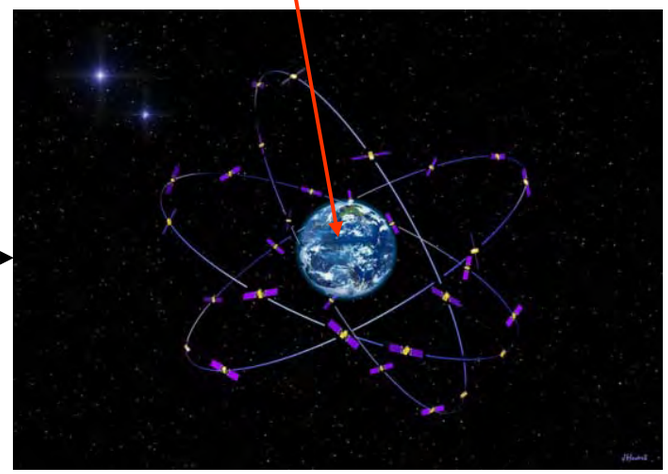
- Photogrammetry from aerial photographs
- Surface generated from sidewalk limits
- Triangular facets
- Precision of vertices

  5 cm planar / 20 cm altitude



Standard approach: Positioning then map-matching

You are here.



GNSS receiver

X, Y, Z, V…

Proprioceptive sensor 1

Proprioceptive sensor 2

Sensor data fusion

λ, ϕ, H

Navigable Map

Map-matching

Position on the map

Relevant attributes of the road segment

You are here, on this segment, at this abscissa

Tightly coupled GNSS/Map localization

GNSS receiver

Raw data

Proprioceptive sensor 1

Proprioceptive sensor 2

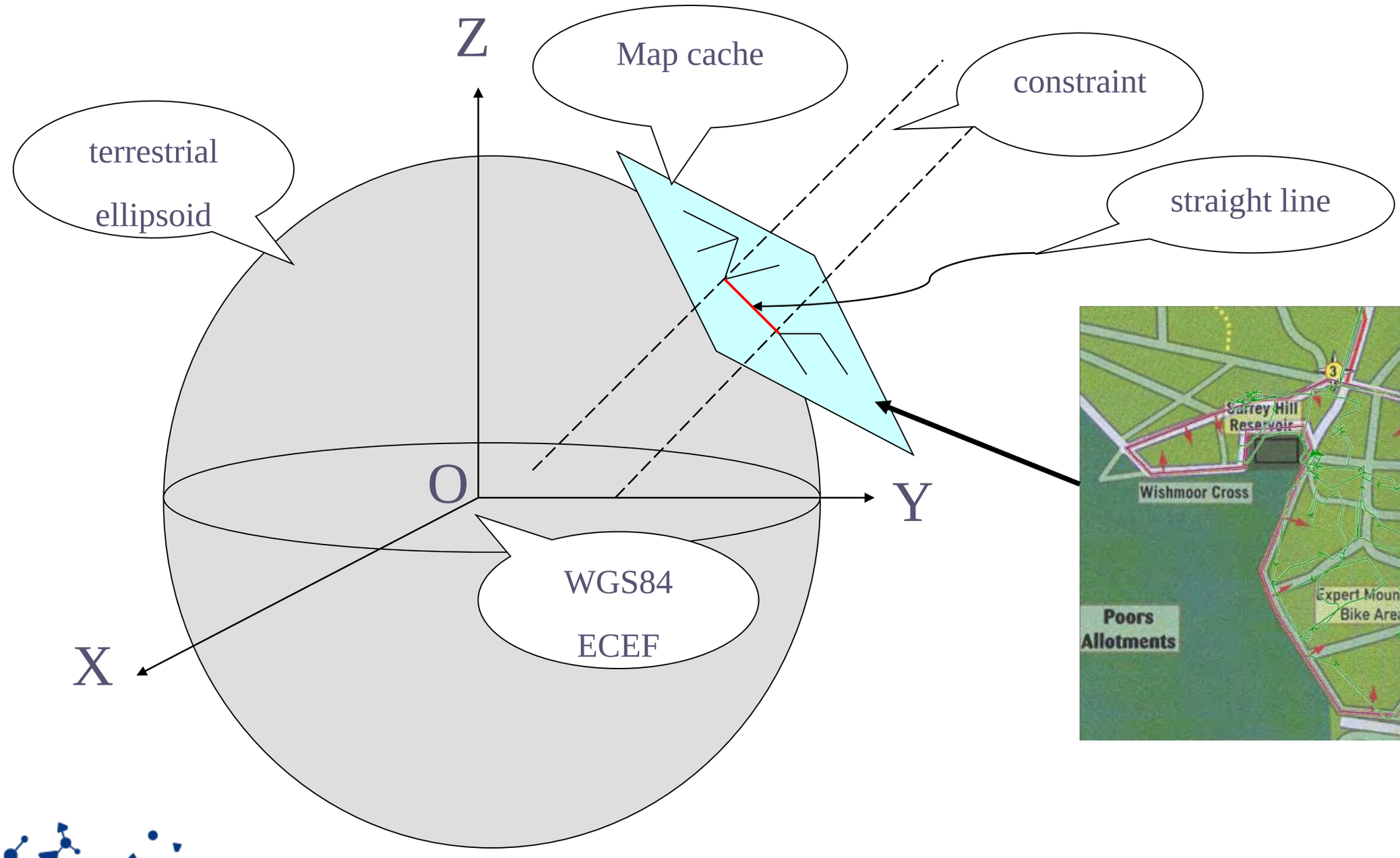
Sensor data fusion and map-matching

Navigable Map

Position on the map

Relevant attributes of the road segment

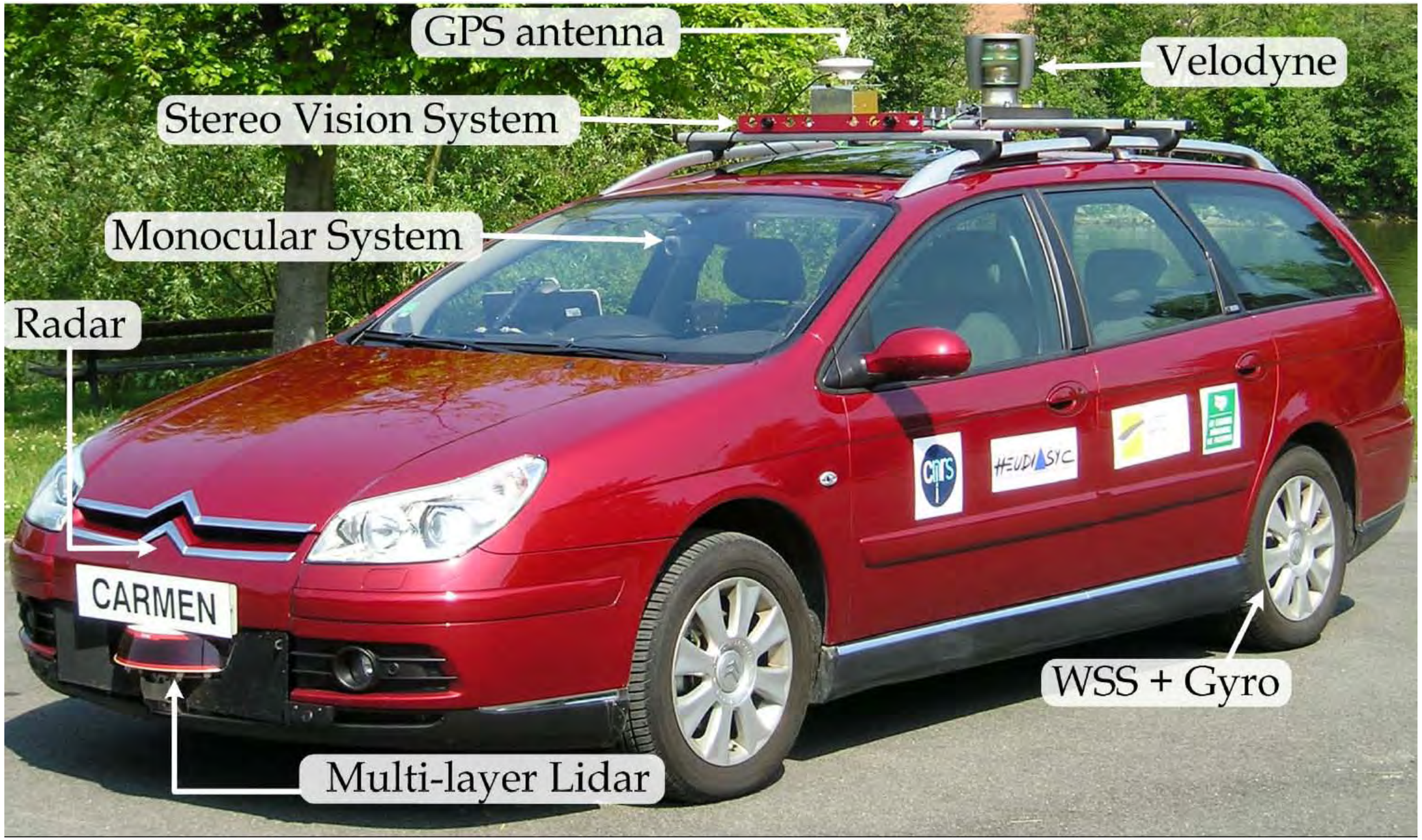
Tightly Coupling GNSS and map data



Bayesian data fusion

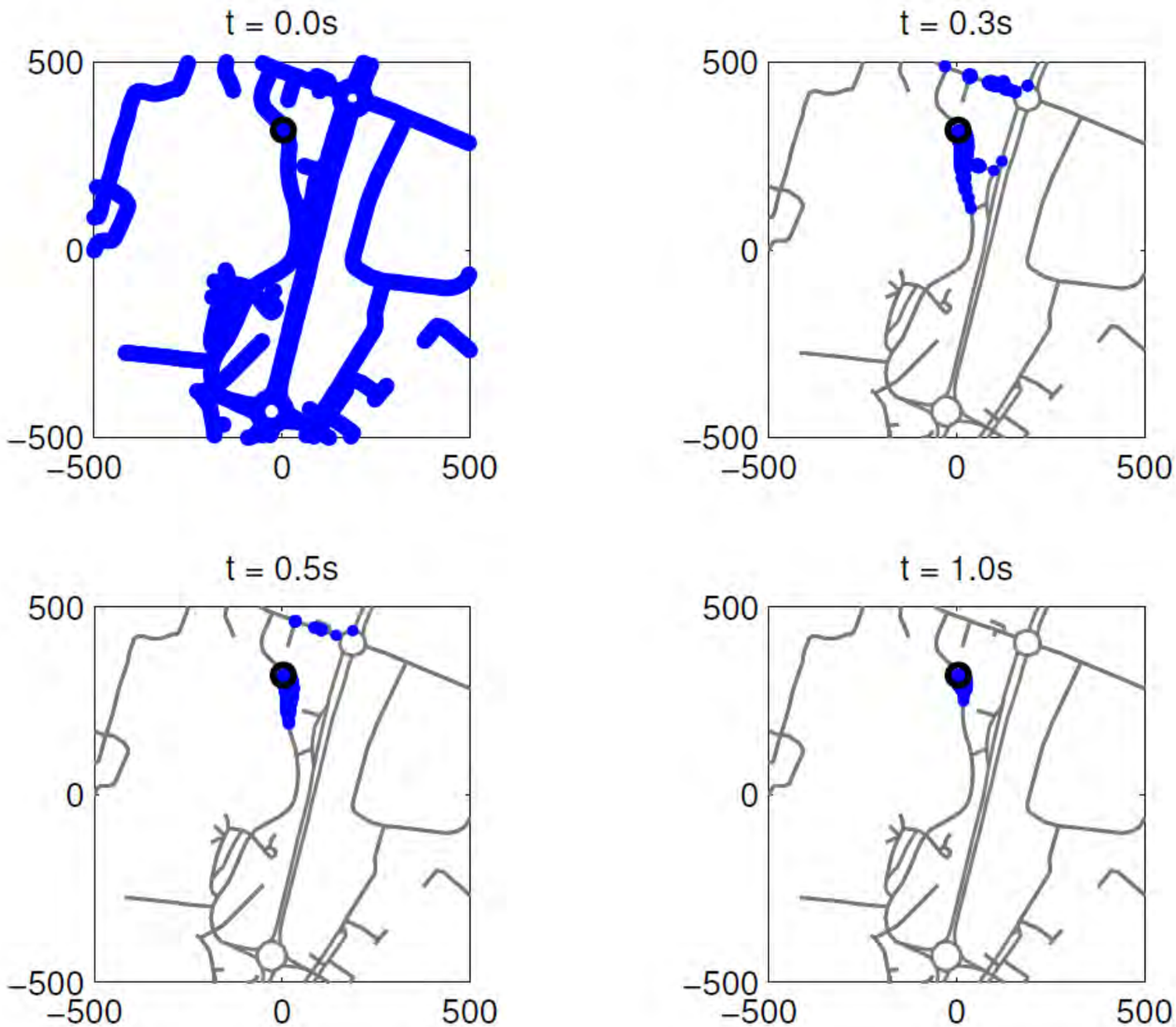
- Static approach
  - Unknown elimination
  - Soft-constraint
- Dynamic approach
  - Soft-constraint: Multi-hypothesis Kalman filtering
  - Hard-constraint: Road tracking using Particle filtering

Experimental vehicle: Carmen

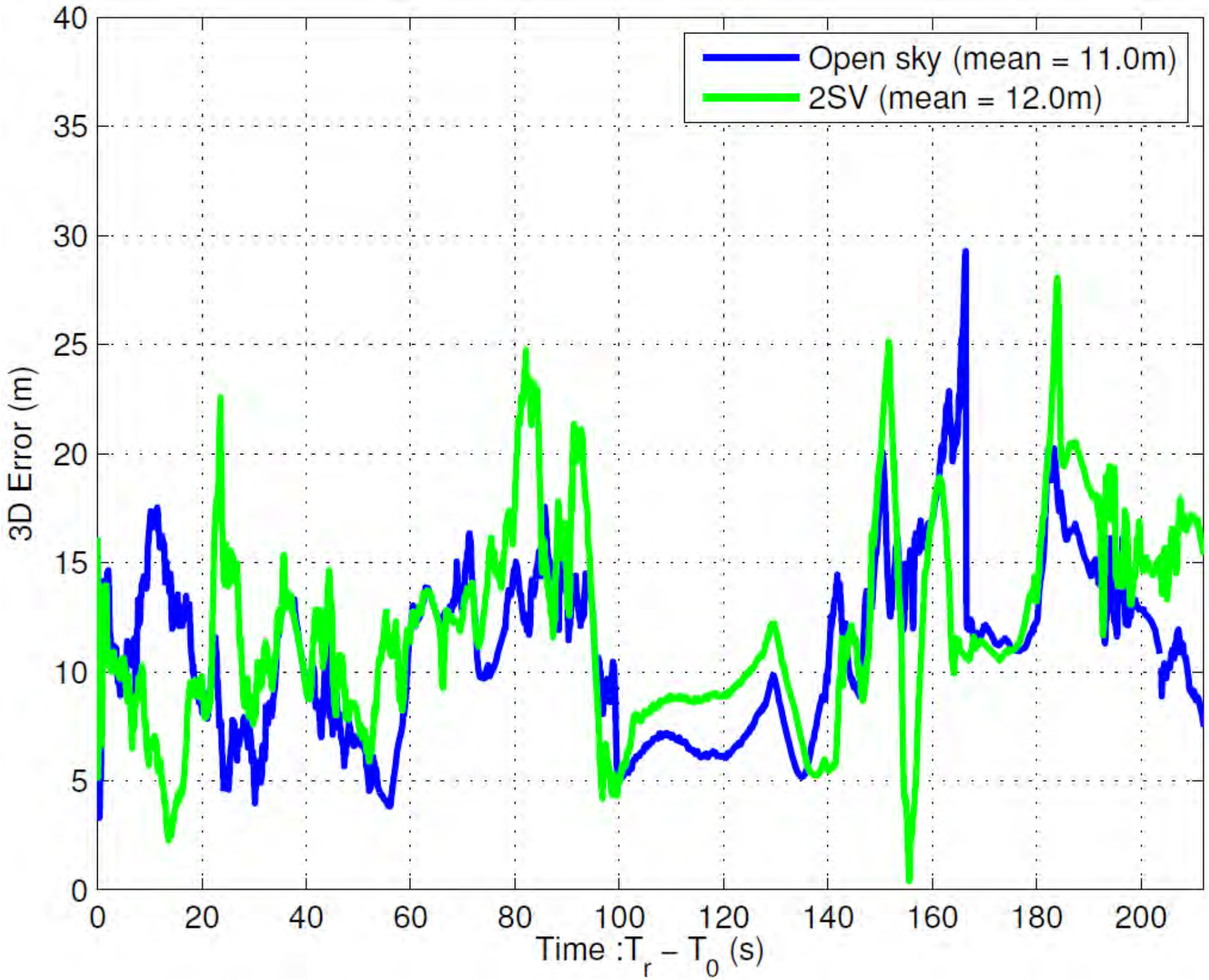


Experimental results

Initialization with 3 satellites



Tracking with two satellites only



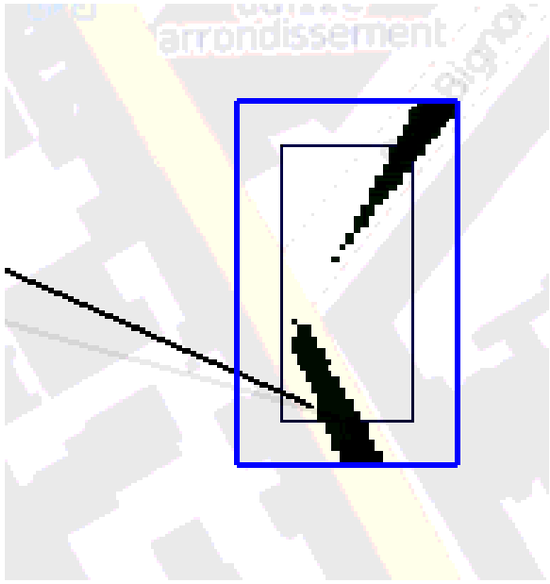
Experimental results

Bounded-error approach

Use of interval analysis

Compute a positioning confidence domain

The set of positions compatible with the measurements and constraints



- Arbitrary shaped solution set
- Disconnected sets in case of ambiguity
- With a integrity risk computed using the pdf of the noise

Bounded-error GPS positioning

Bounded-error framework

- Measurements = Intervals
- Intervals are assumed to include the true value with a given probability

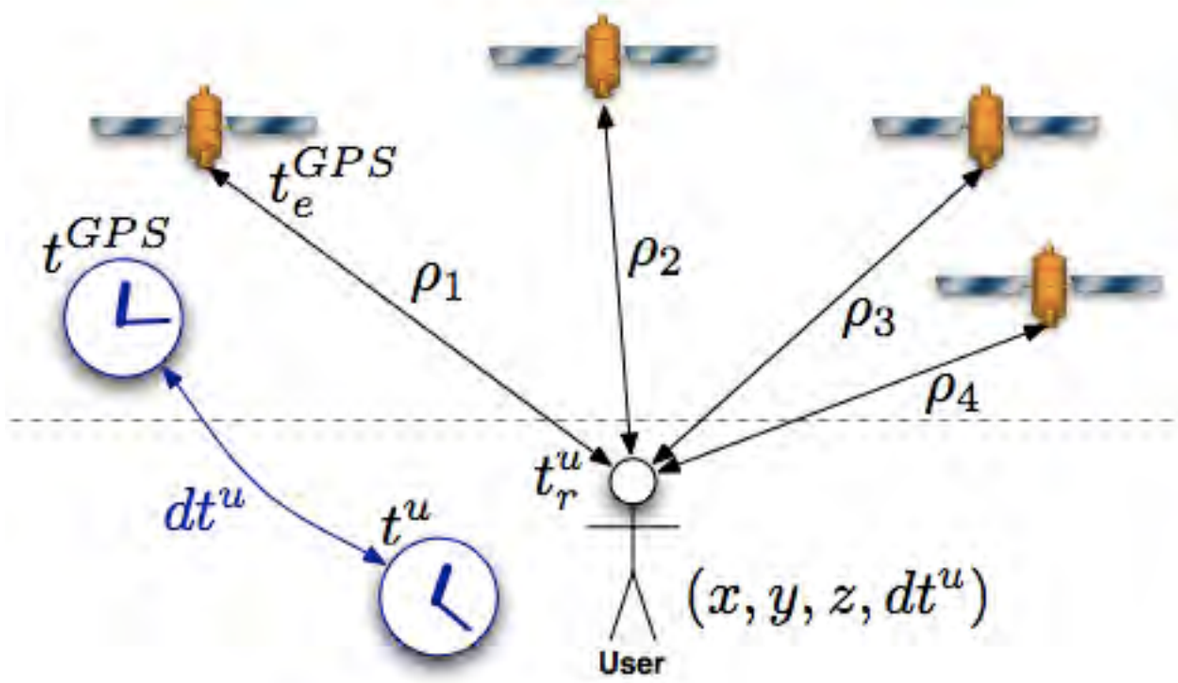
Positioning is a Constraint Satisfaction Problem

- Measurements = Constraints on position
- Position = Intersection of constraints

Bounded-error GPS positioning

Receiver measures pseudo-ranges:
range + offset

4 unknowns: x, y, z, dtu -> 4-D boxes



Pseudo-range observation model:

$$\begin{cases} \rho_1 = \sqrt{(x-x_{s1})^2+(y-y_{s1})^2+(z-z_{s1})^2} + c \cdot dtu \\ \rho_2 = \sqrt{(x-x_{s2})^2+(y-y_{s2})^2+(z-z_{s2})^2} + c \cdot dtu \\ \quad \ldots \\ \rho_p = \sqrt{(x-x_{sp})^2+(y-y_{sp})^2+(z-z_{sp})^2} + c \cdot dtu \end{cases}$$

$x_{s_i}, y_{s_i}, z_{s_i}$ are satellite positions (broadcast)

ρ_i are corrected pseudoranges:

Pseudorange constraint

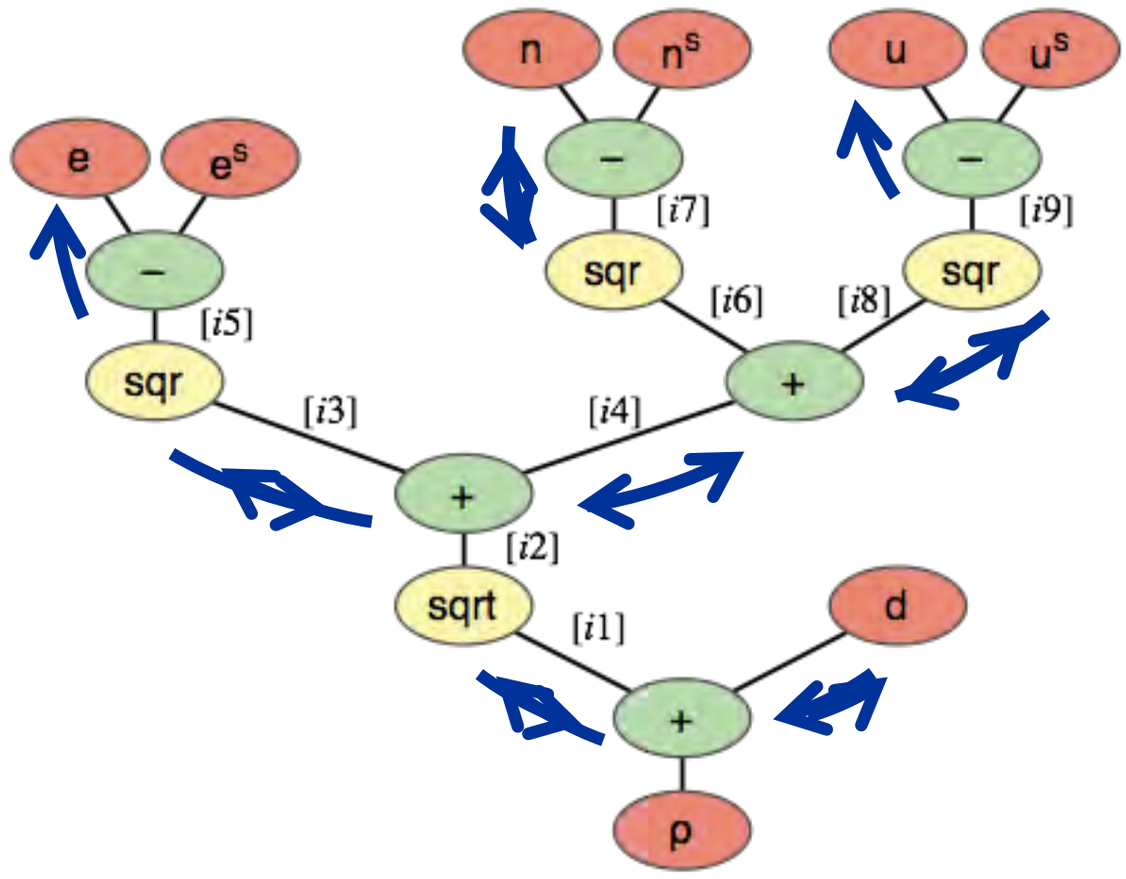
Each measurement is a constraint on position

$$[\rho_i] = \sqrt{([e] - [e_i^s])^2 + ([n] - [n_i^s])^2 + ([u] - [u_i^s])^2} + [d]$$

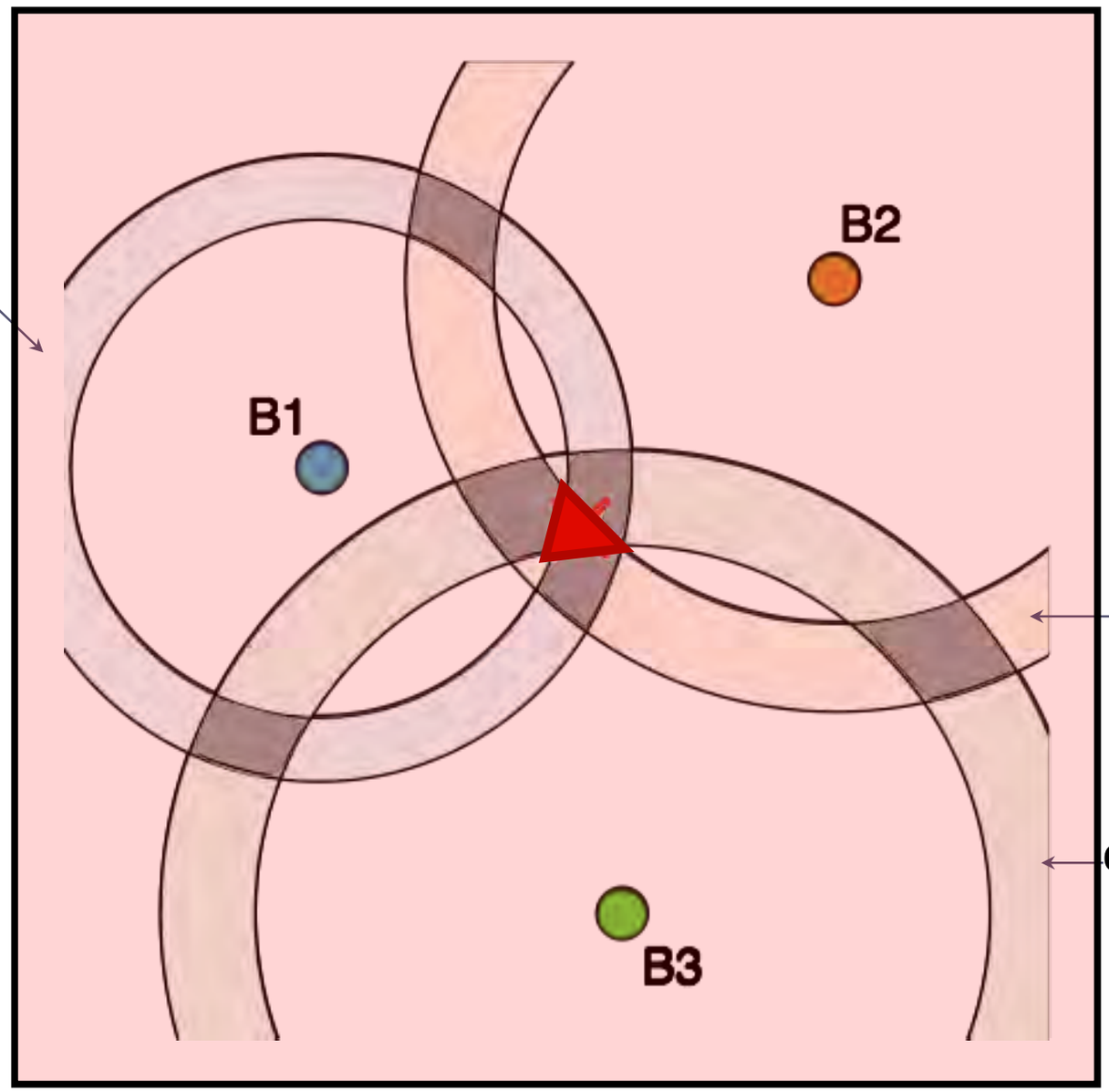
Prior position box is contracted with « fall - climb » constraint propagation

The domains of the variables are narrowed without losing solution

Contraction is successively applied with each pseudo-range, until a fixed point

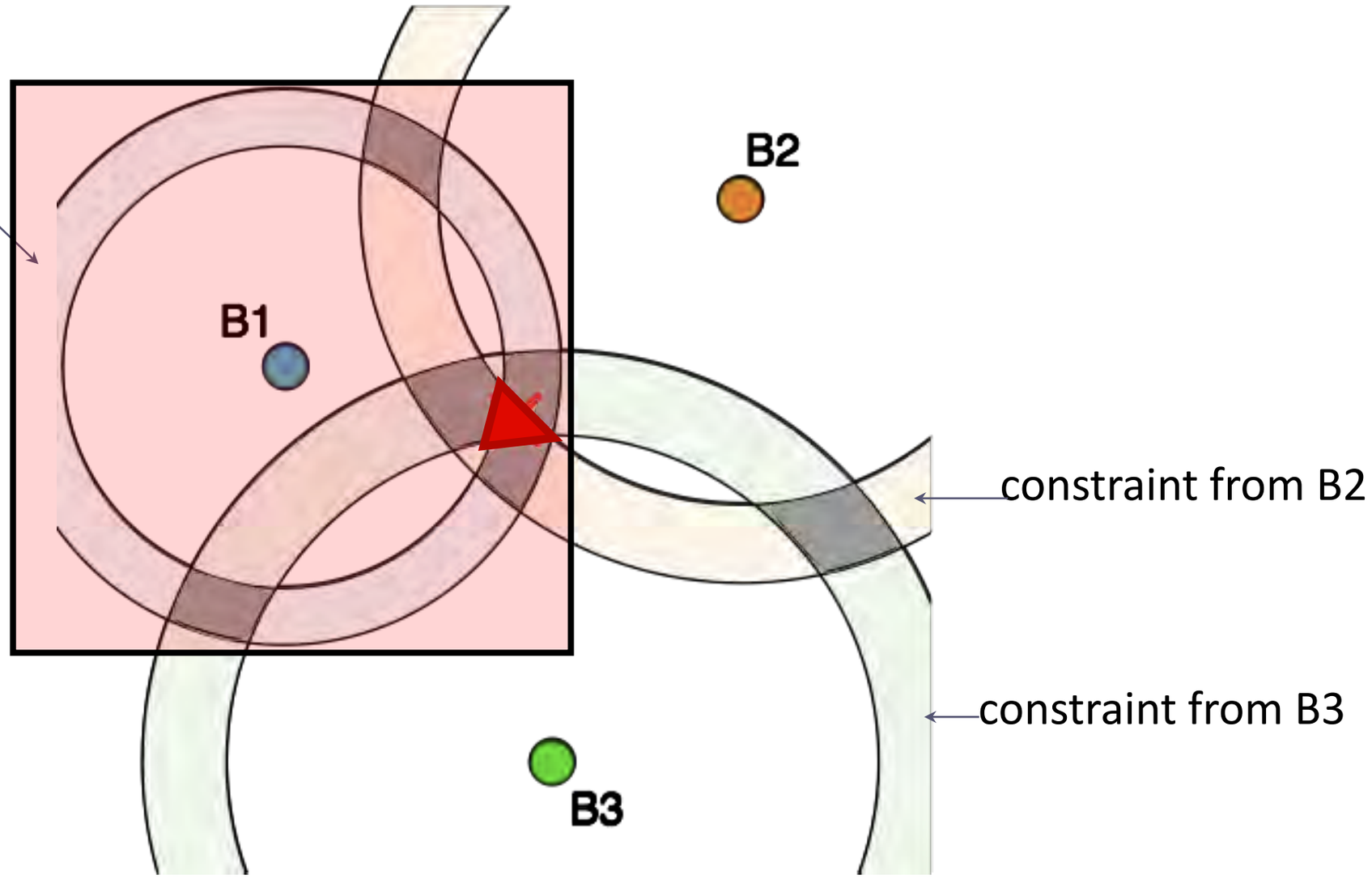


Simplified 2D example



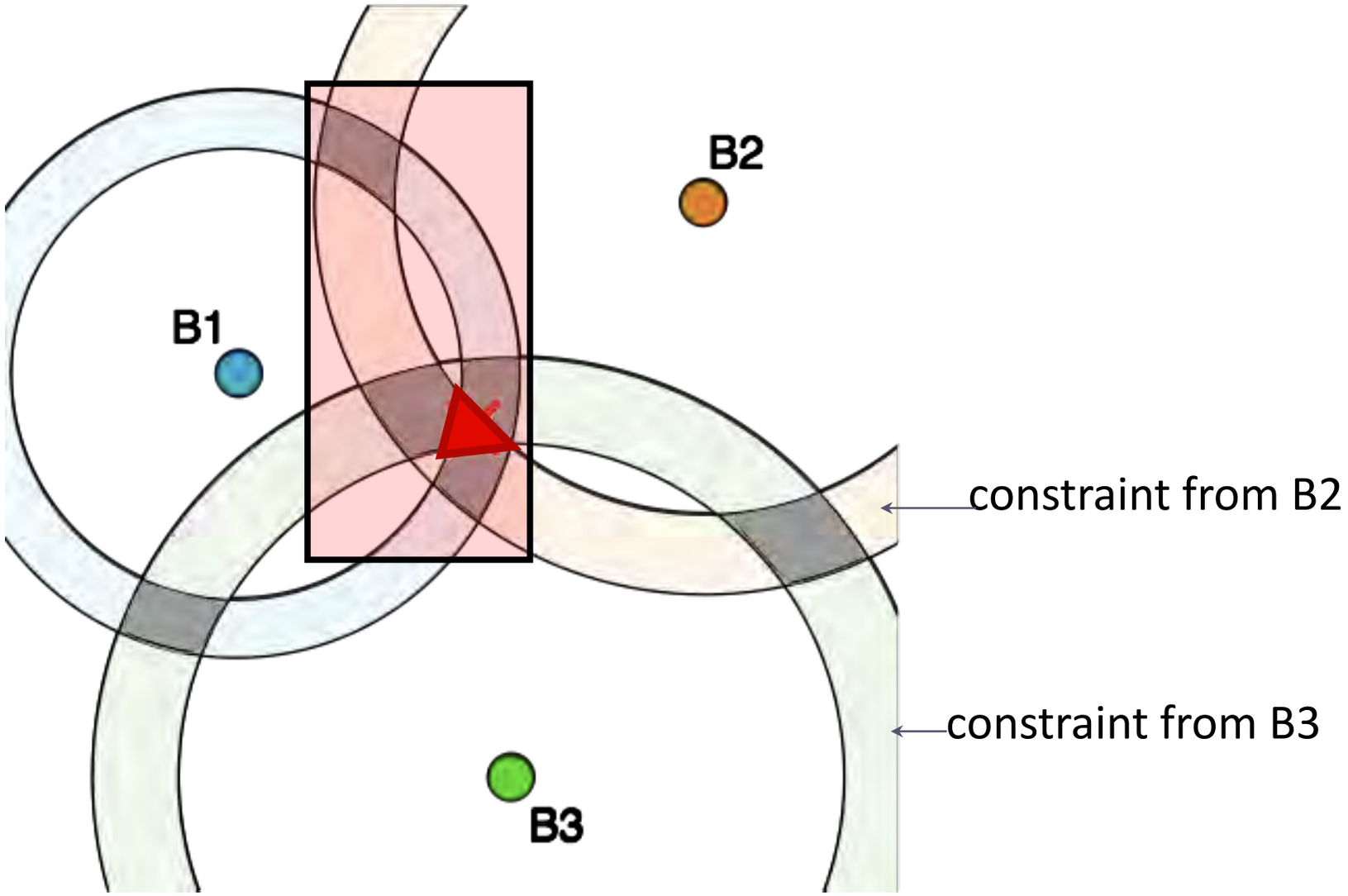
Simplified 2D example

constraint from B1
range measurement



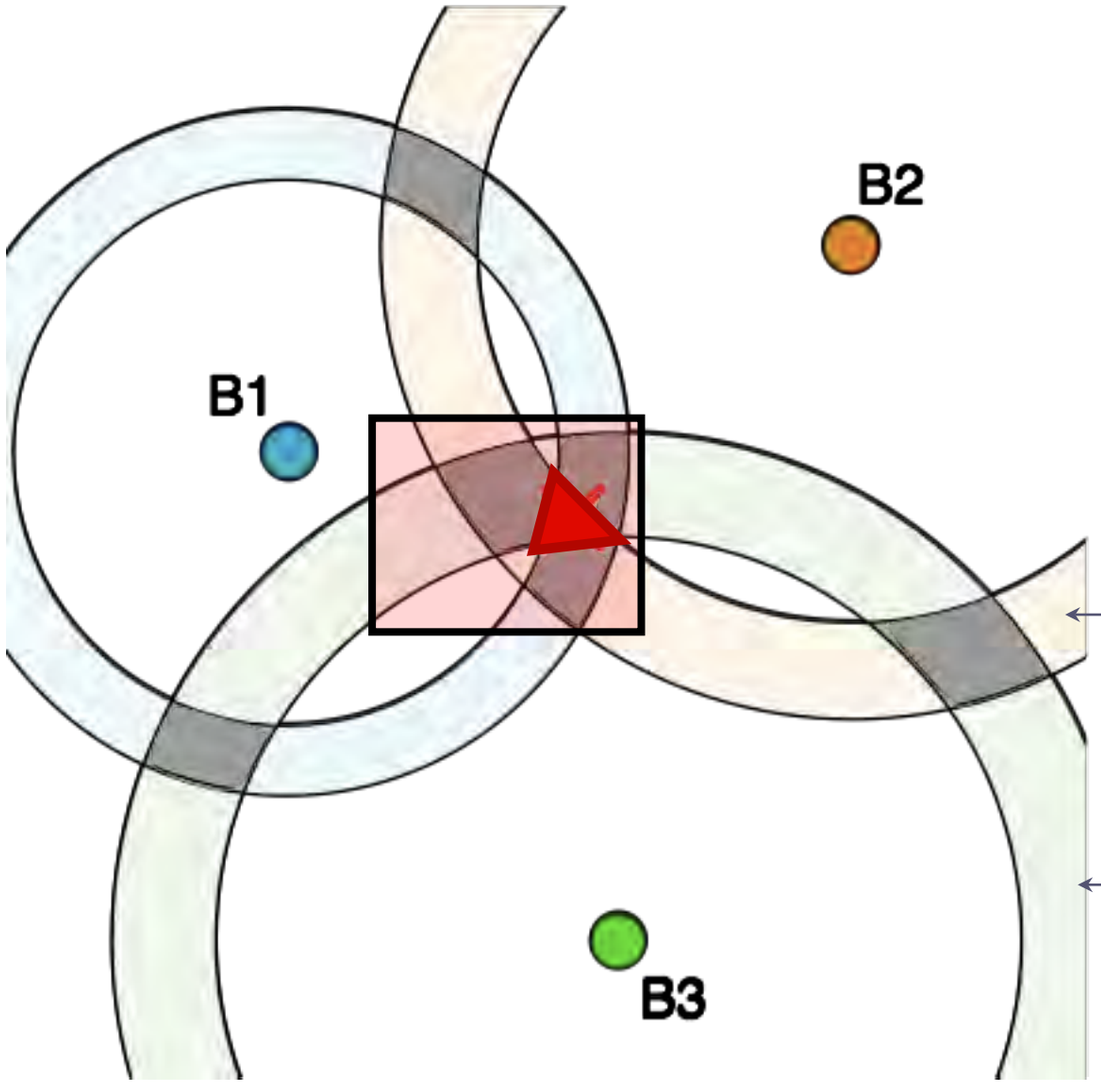
Simplified 2D example

constraint from B1
range measurement



Simplified 2D example

constraint from B1
range measurement

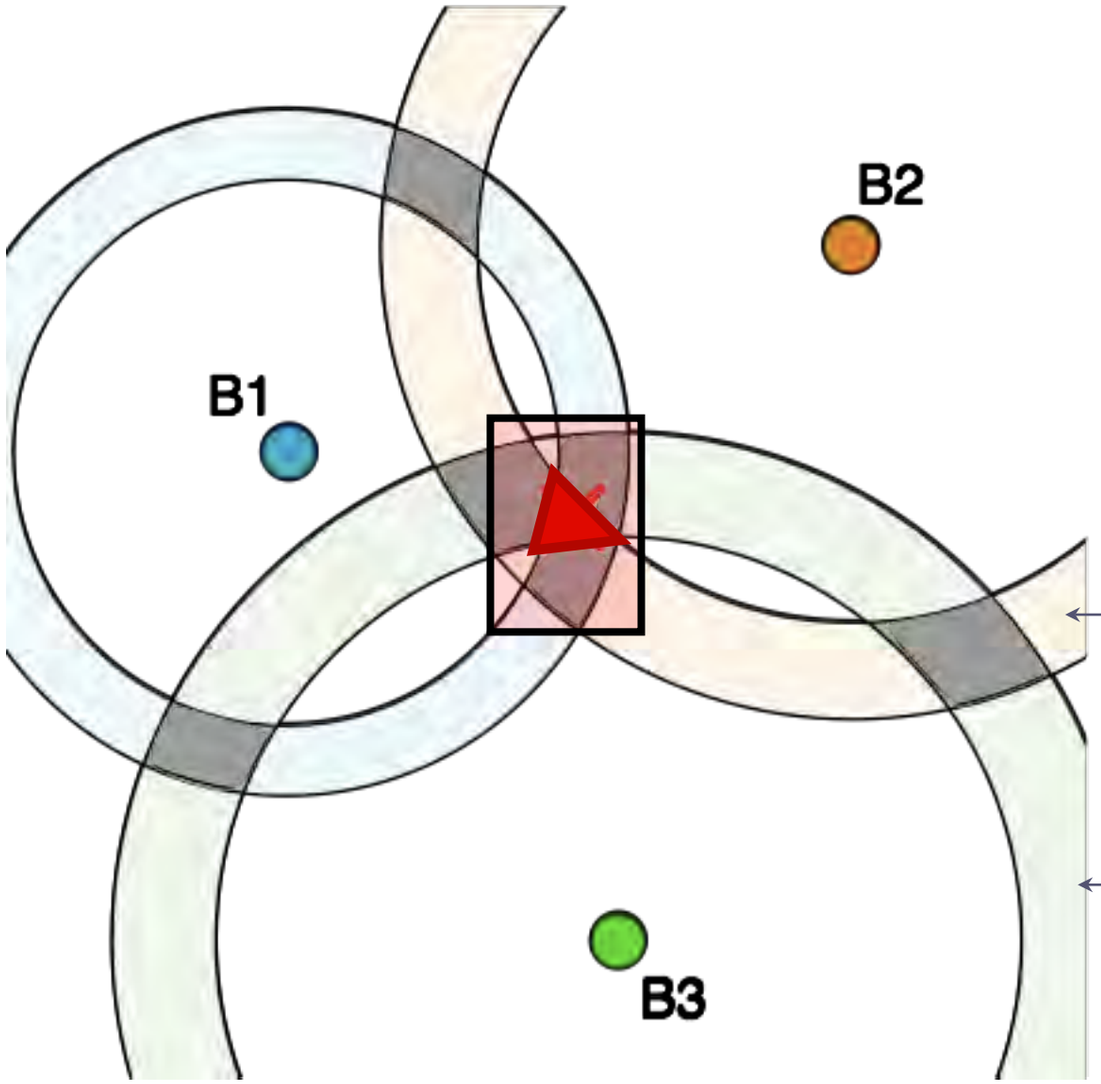


constraint from B2

constraint from B3

Simplified 2D example

constraint from B1
range measurement

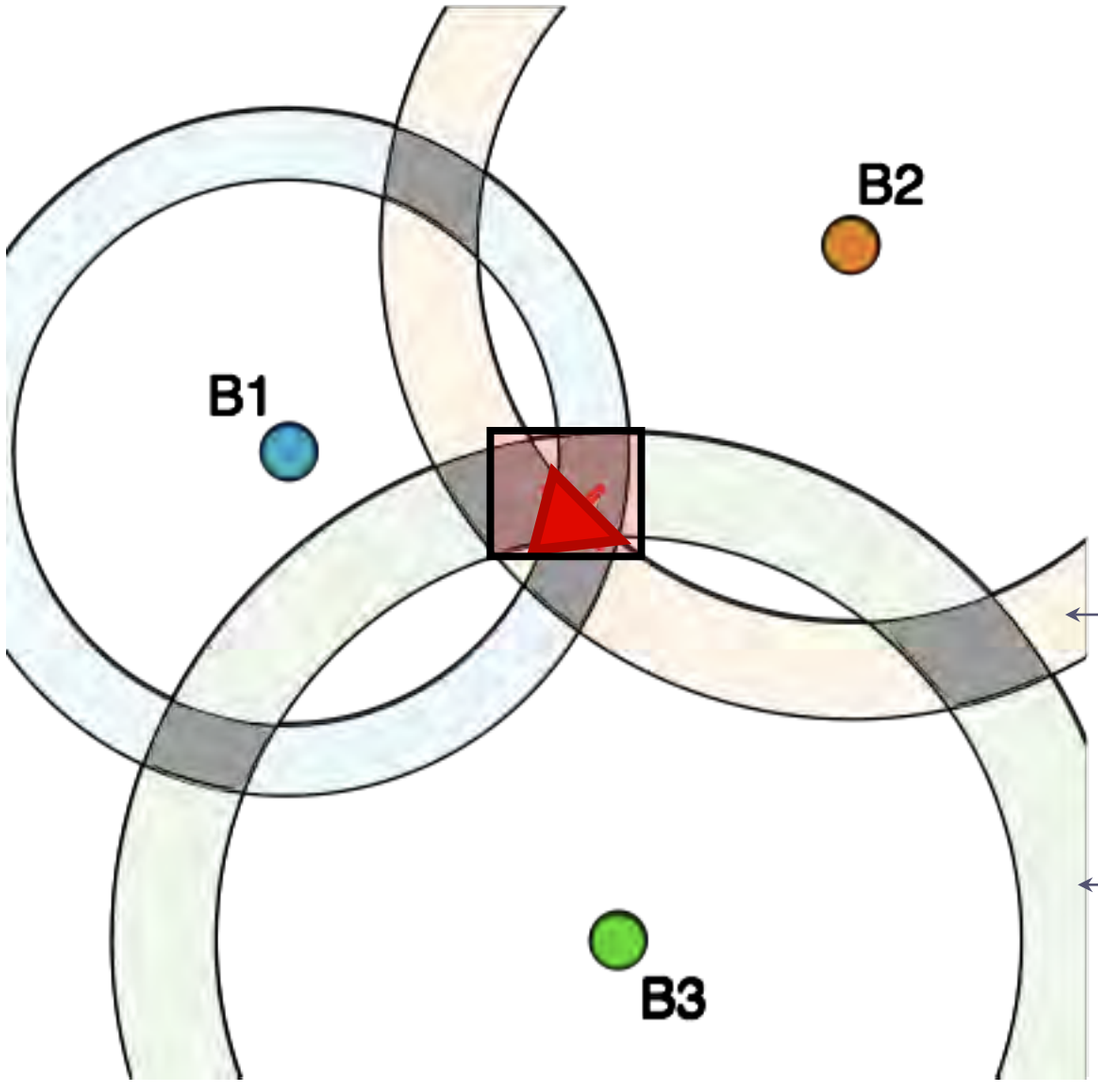


constraint from B2

constraint from B3

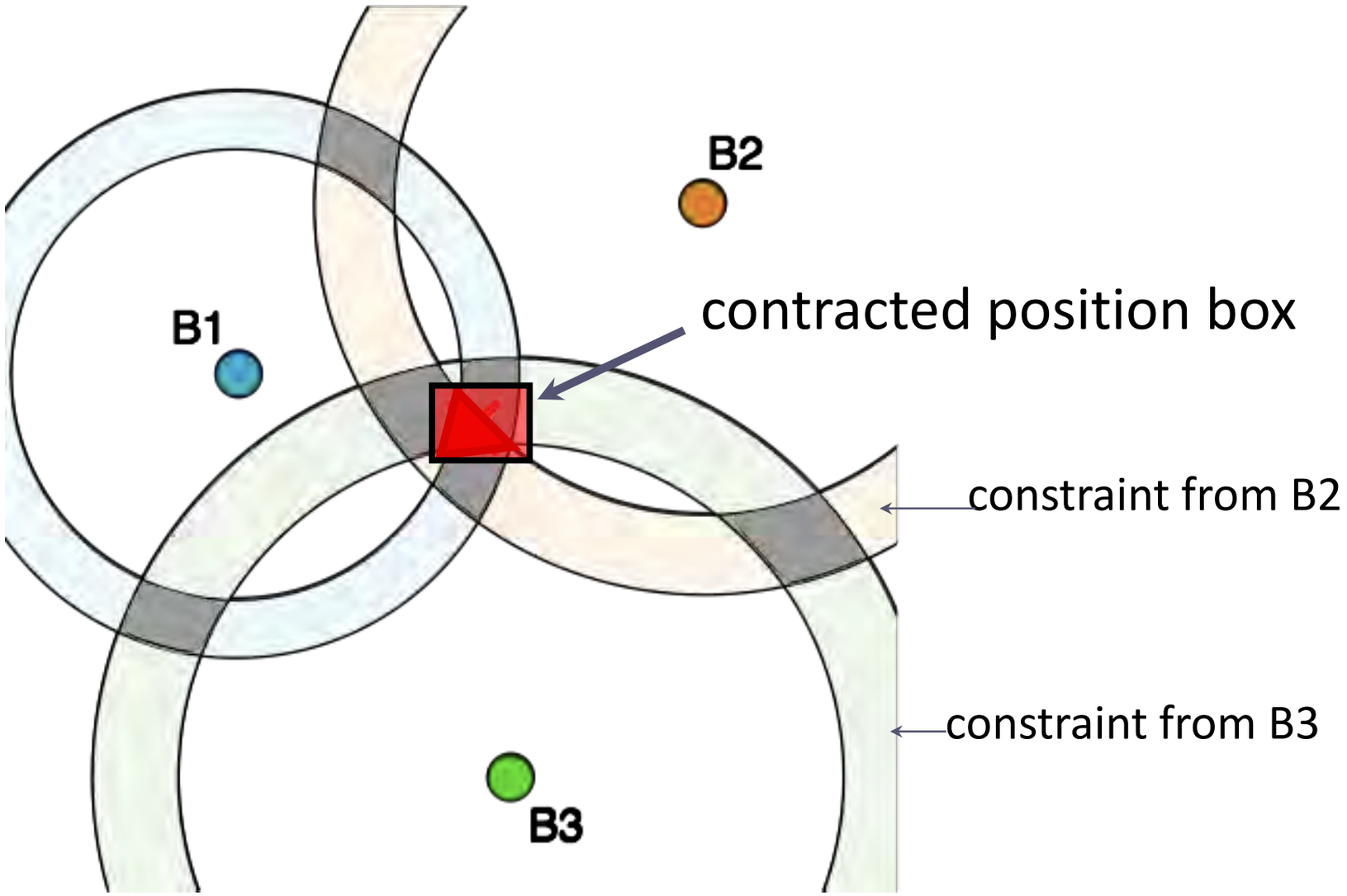
Simplified 2D example

constraint from B1
range measurement



Simplified 2D example

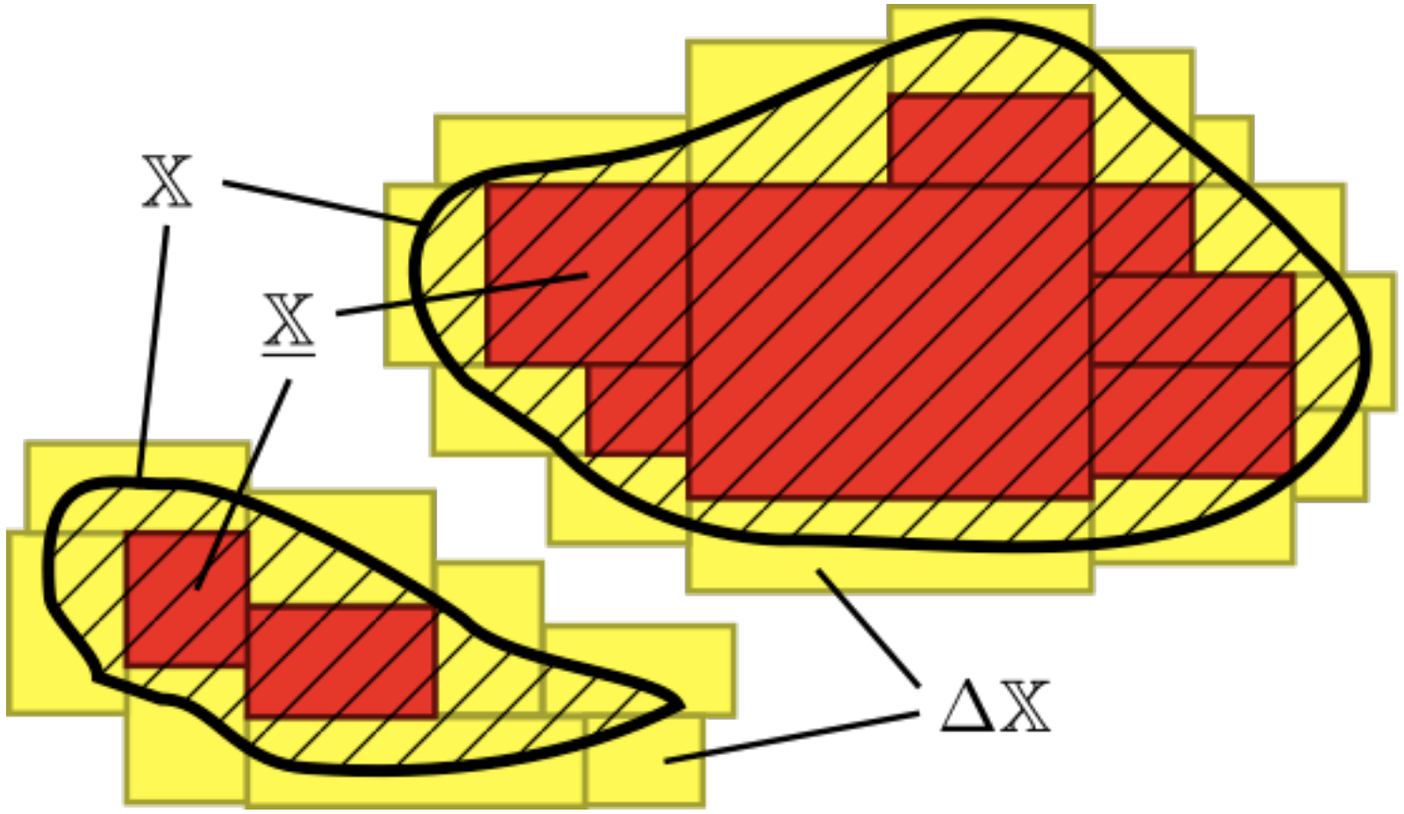
constraint from B1
range measurement



Subpavings

Boxes only provide a rough approximation

Better approximation of arbitrary sets: subpavings



3D facets constraint

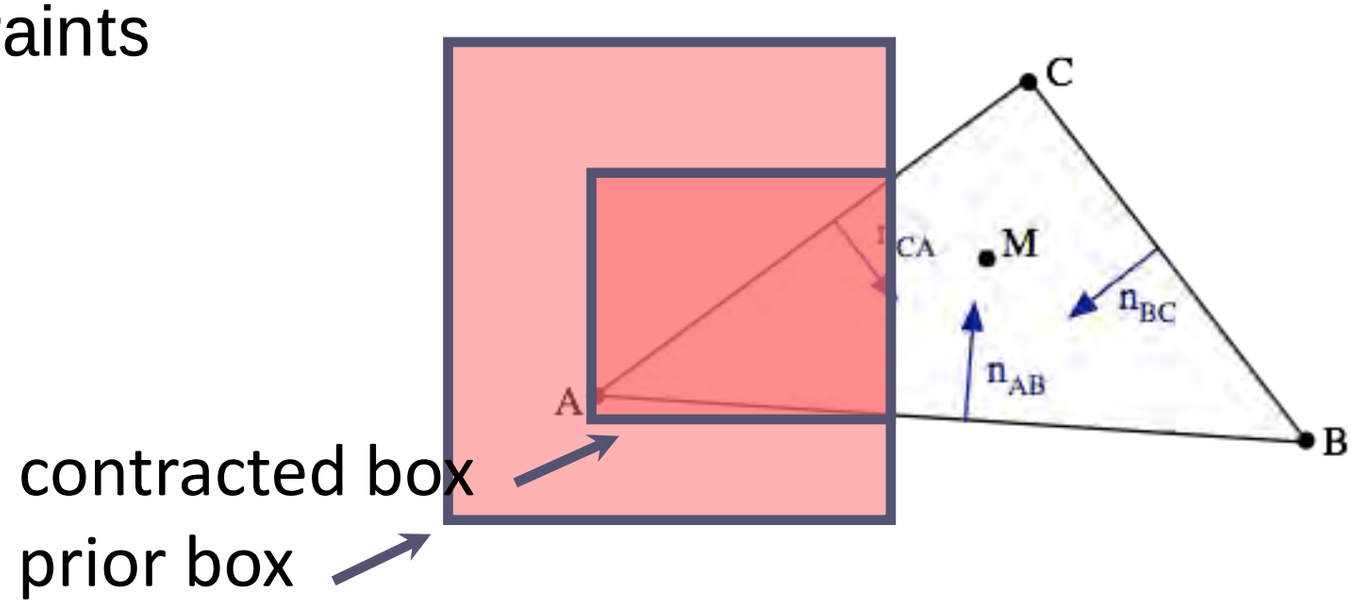
1 facet constraint

- Vertices coordinates are boxes (uncertainty)
- Facet plane constraint

$$(\overrightarrow{AB} \wedge \overrightarrow{AC}) \cdot \overrightarrow{AM} = 0$$

- 3 facet edges constraints

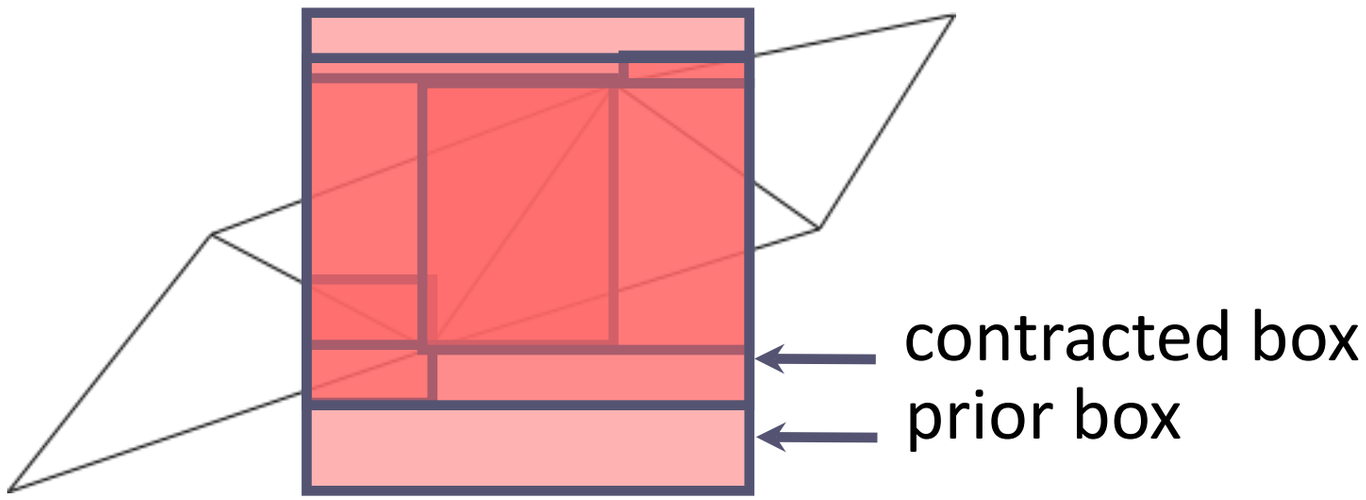
$$\overrightarrow{AM} \cdot \overrightarrow{n_{AB}} \geq 0$$
$$\overrightarrow{BM} \cdot \overrightarrow{n_{BC}} \geq 0$$
$$\overrightarrow{CM} \cdot \overrightarrow{n_{CA}} \geq 0$$



3D facets constraint

Drivable space constraint

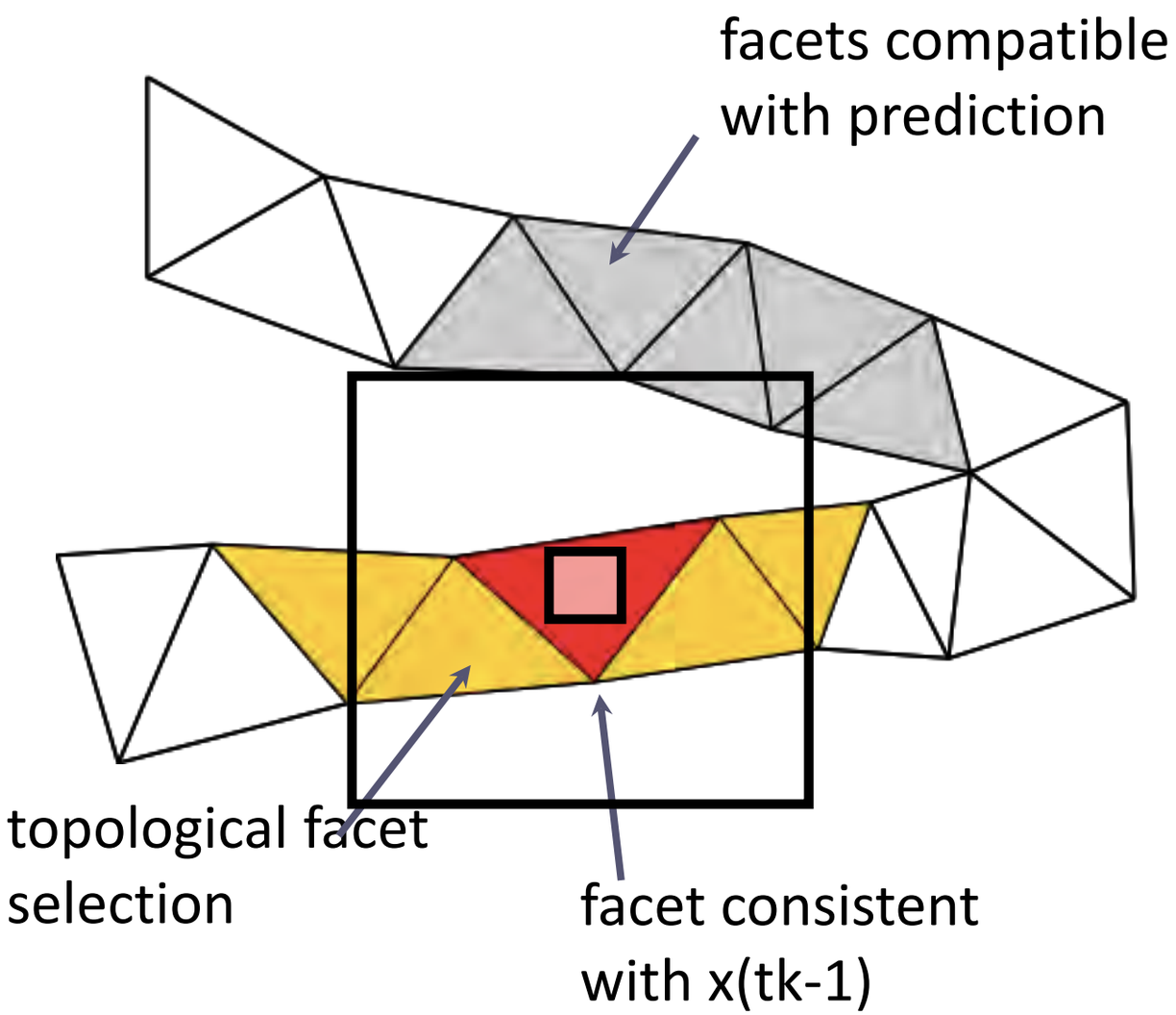
- Union of facet constraints



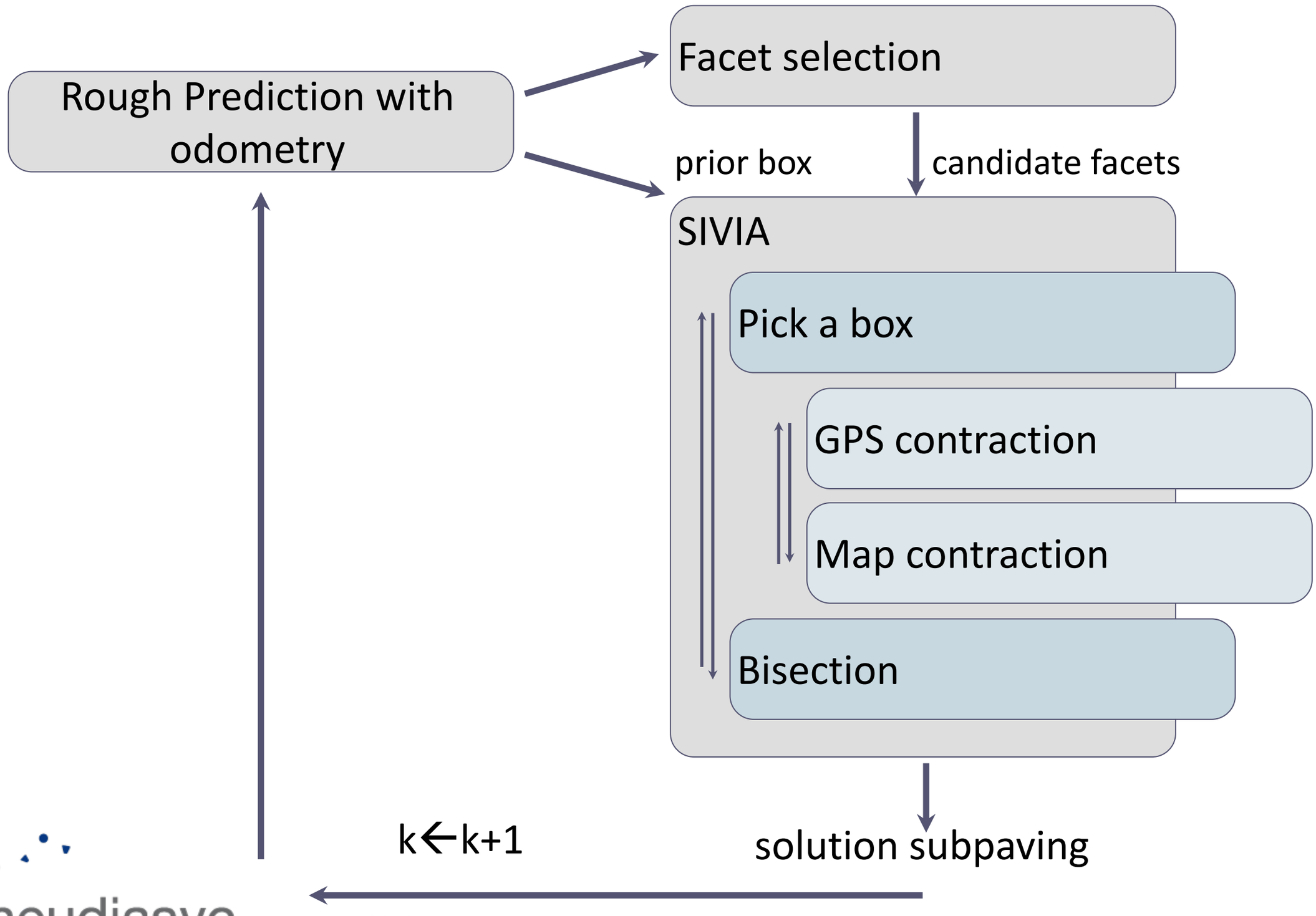
Facet selection (map matching)

Use topology to mark eligible neighbors from previous epoch facets set.

- Speeds up computation
- Limits ambiguous solutions in poor GPS conditions and dense road networks



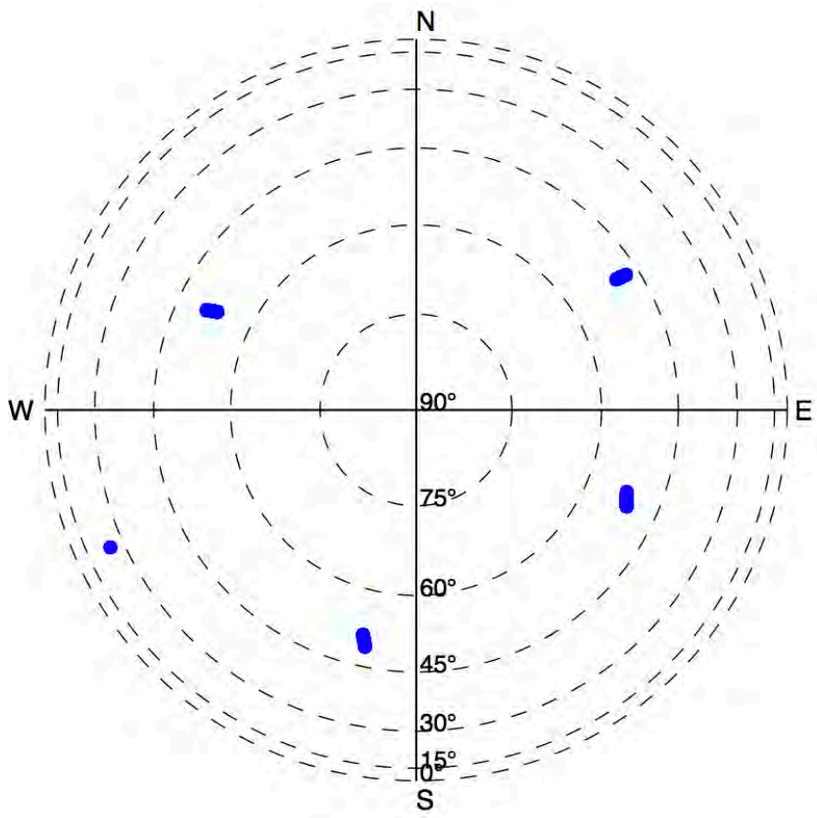
Positioning algorithm

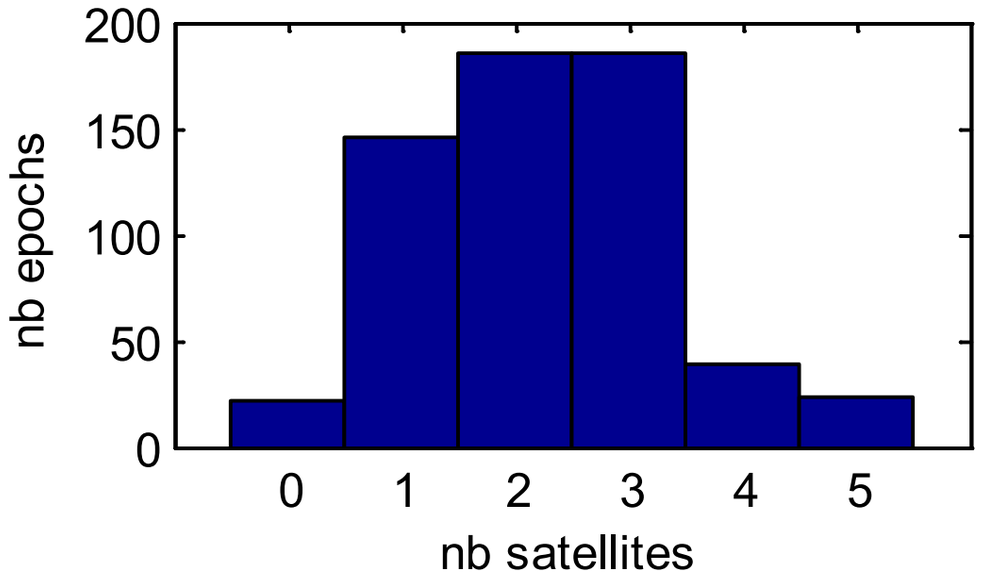


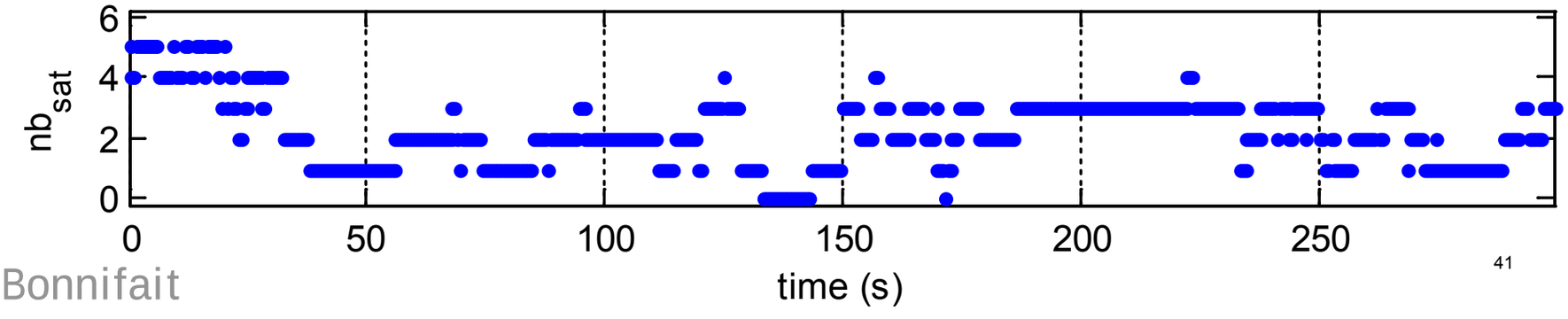
Experiment

12th arrondissement Mairie in Paris

Septentrio PolaRx, SNR threshold of 35dBHz







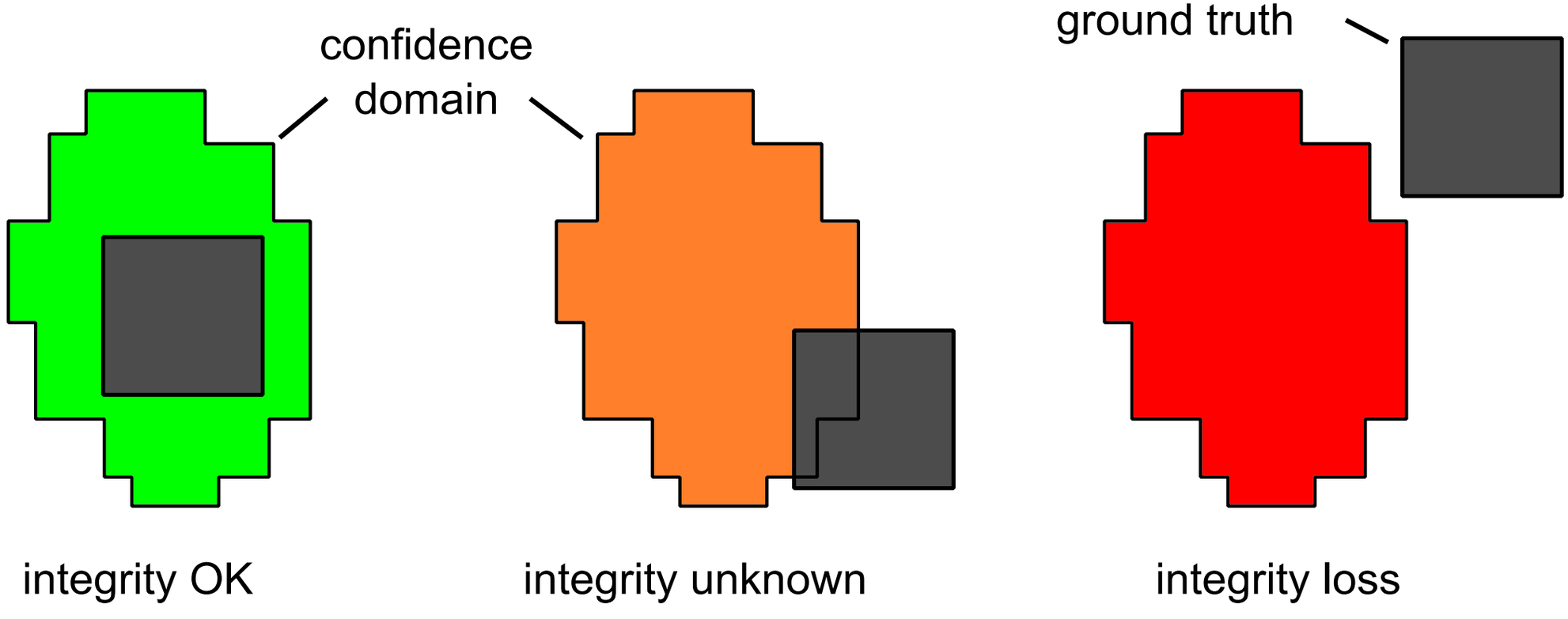

Rue Pleyel
Rue Dudorufaut
Rue Duguommier
Boulevard de Berry
Metro 6

Evaluation methodology

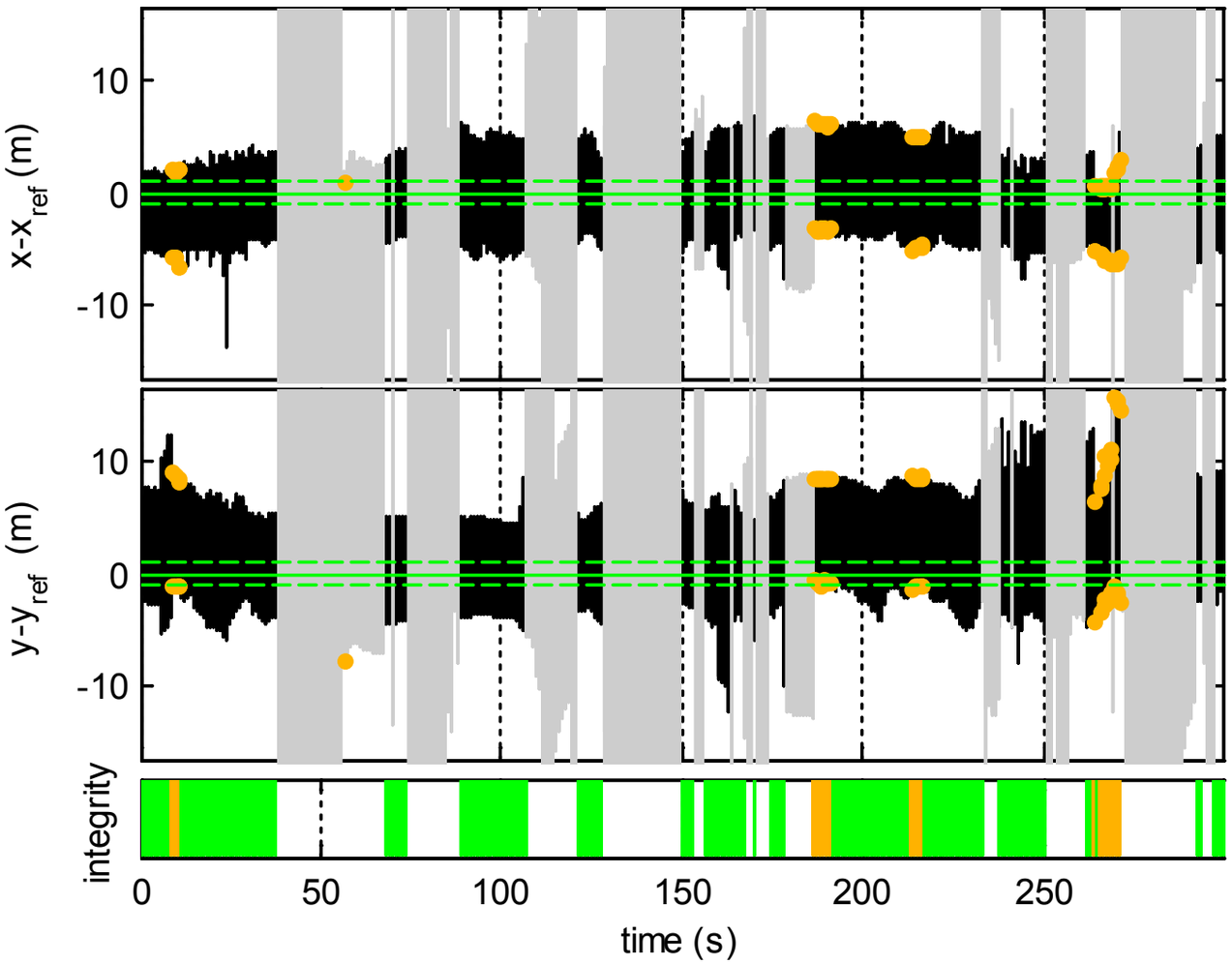
Availability

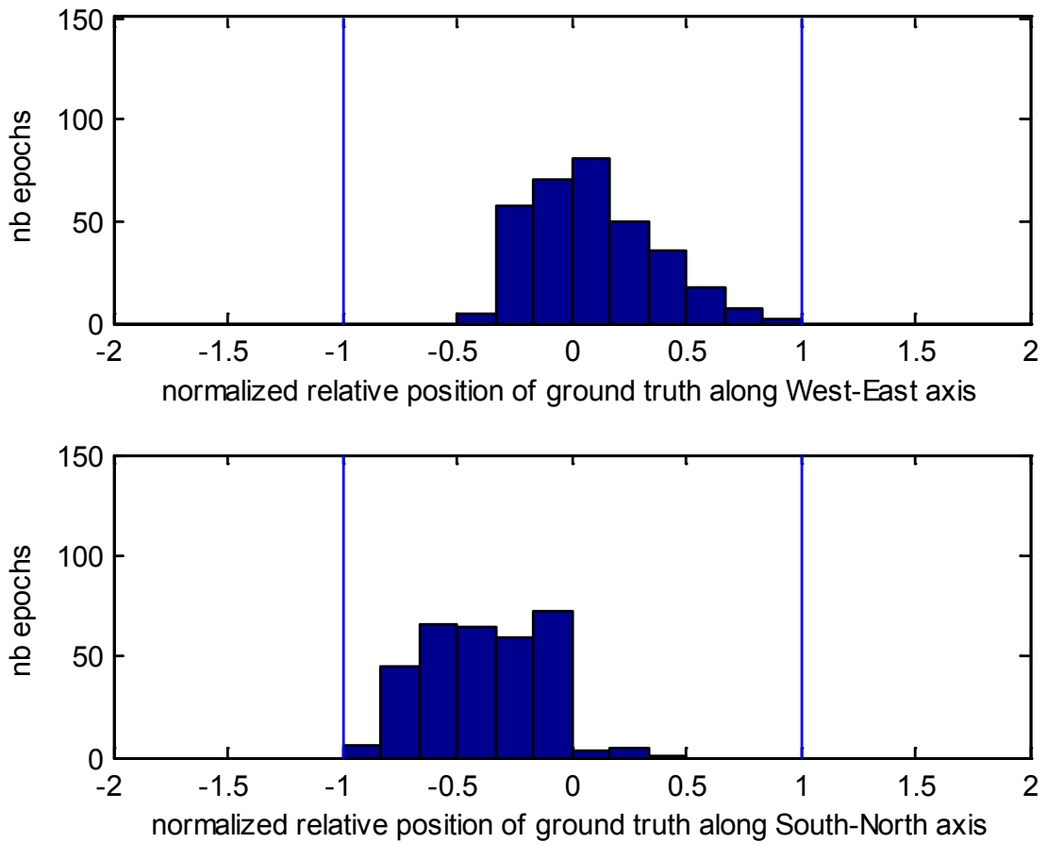
- bounding box of the sub-paving
- its radius is compared with a 10-meter Alarm Limit (HAL)

Integrity validation of the solution



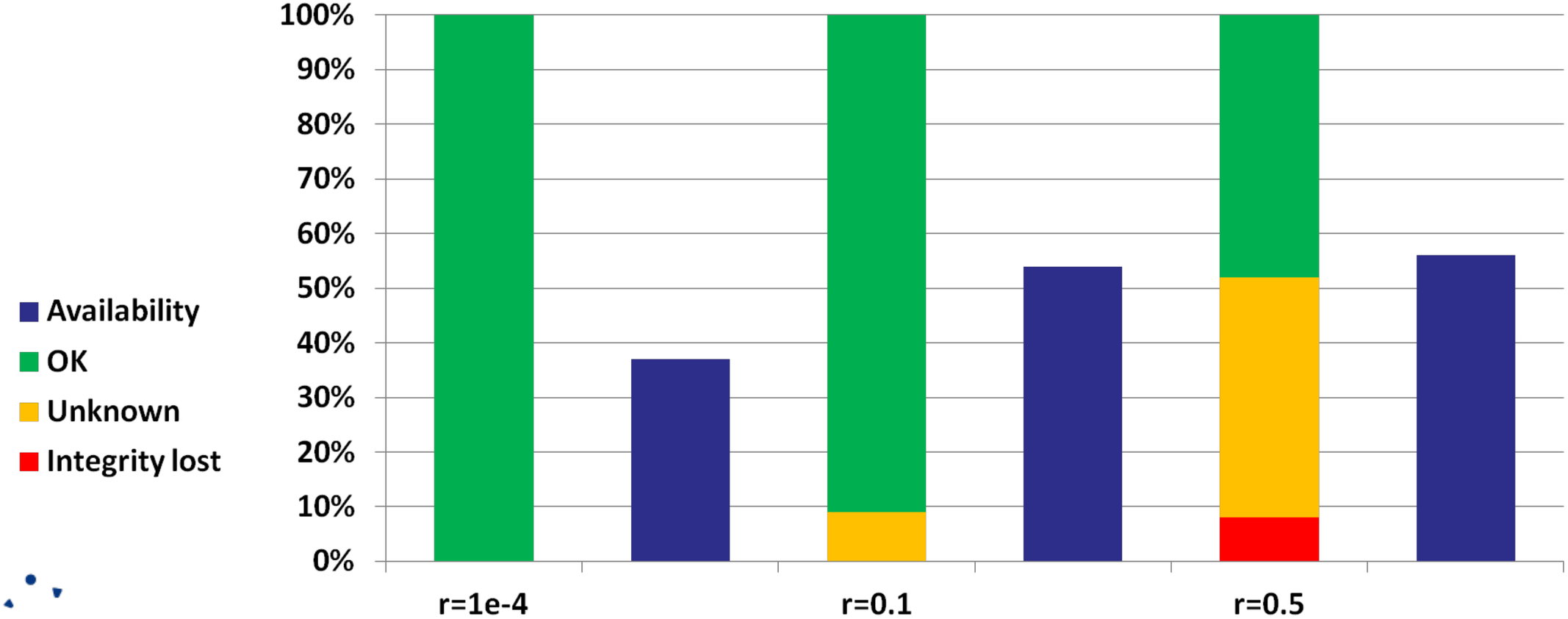
Results with a 10% risk





Availability and integrity statistics

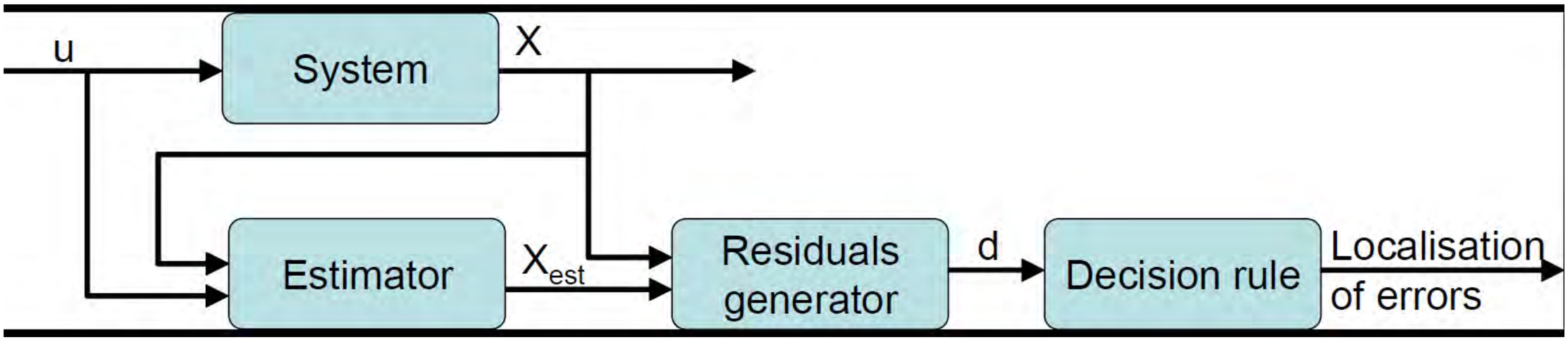
Integrity risk	r=0.01%	r=10%	r=50%
Availability	37%	54%	56%
Integrity OK	100%	91%	48%
Integrity unknown	0%	9%	44%
Integrity lost	0%	0%	8%



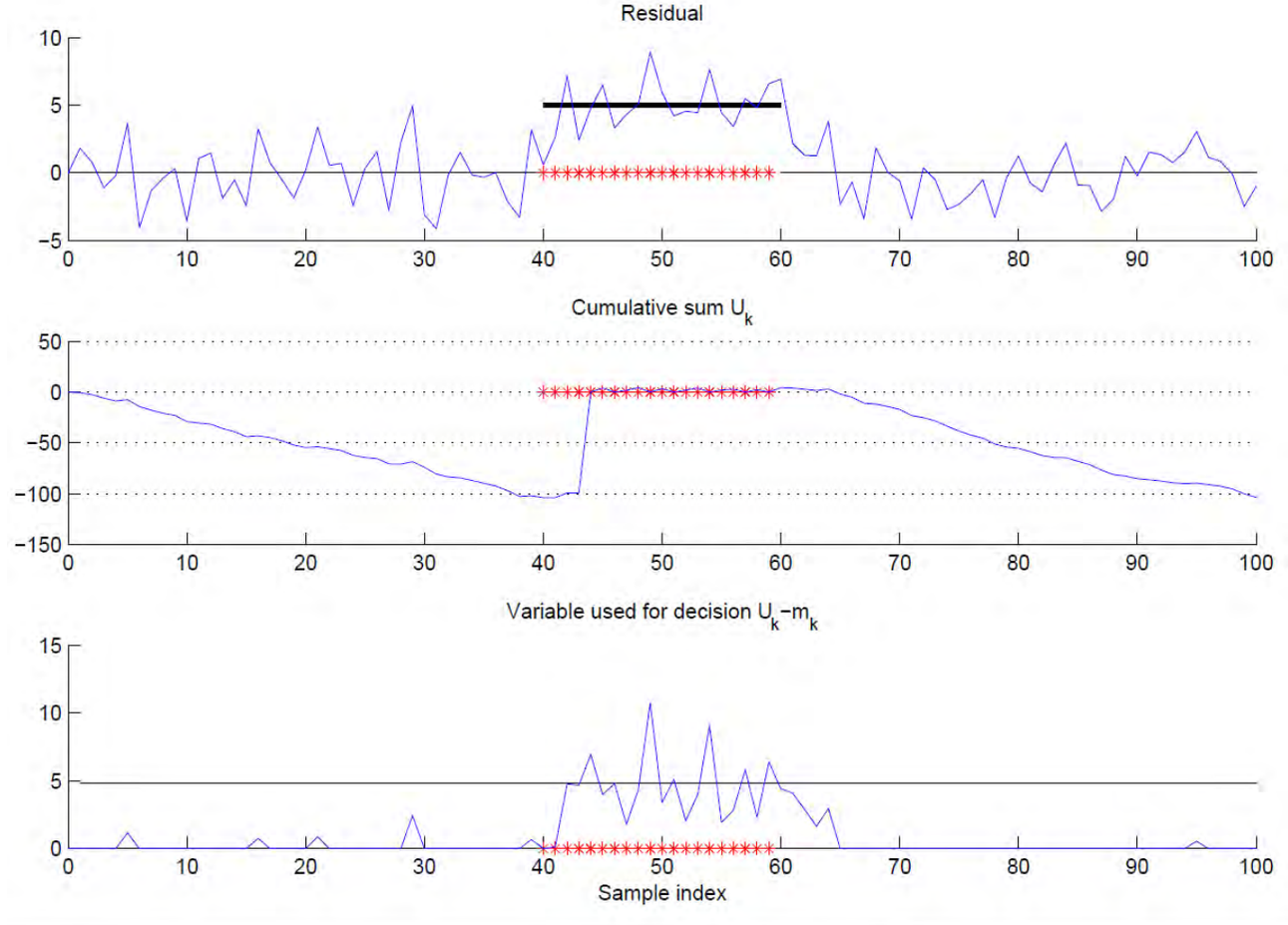
Integrity of road maps for ADAS

- Maps are always out-of-date
  - Evolution of the road network
  - Errors in mapping process

Error detection and localization

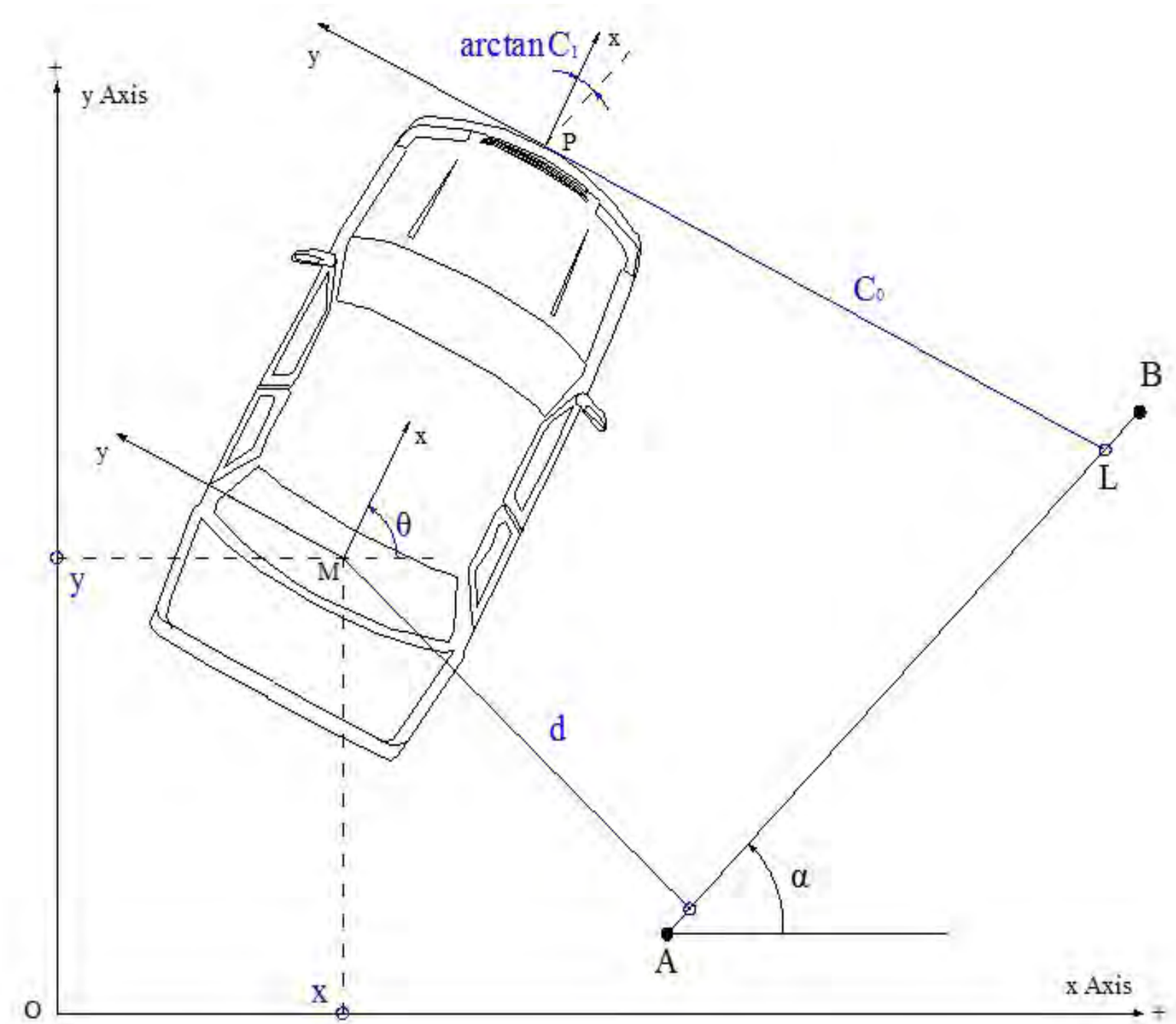


Page's test

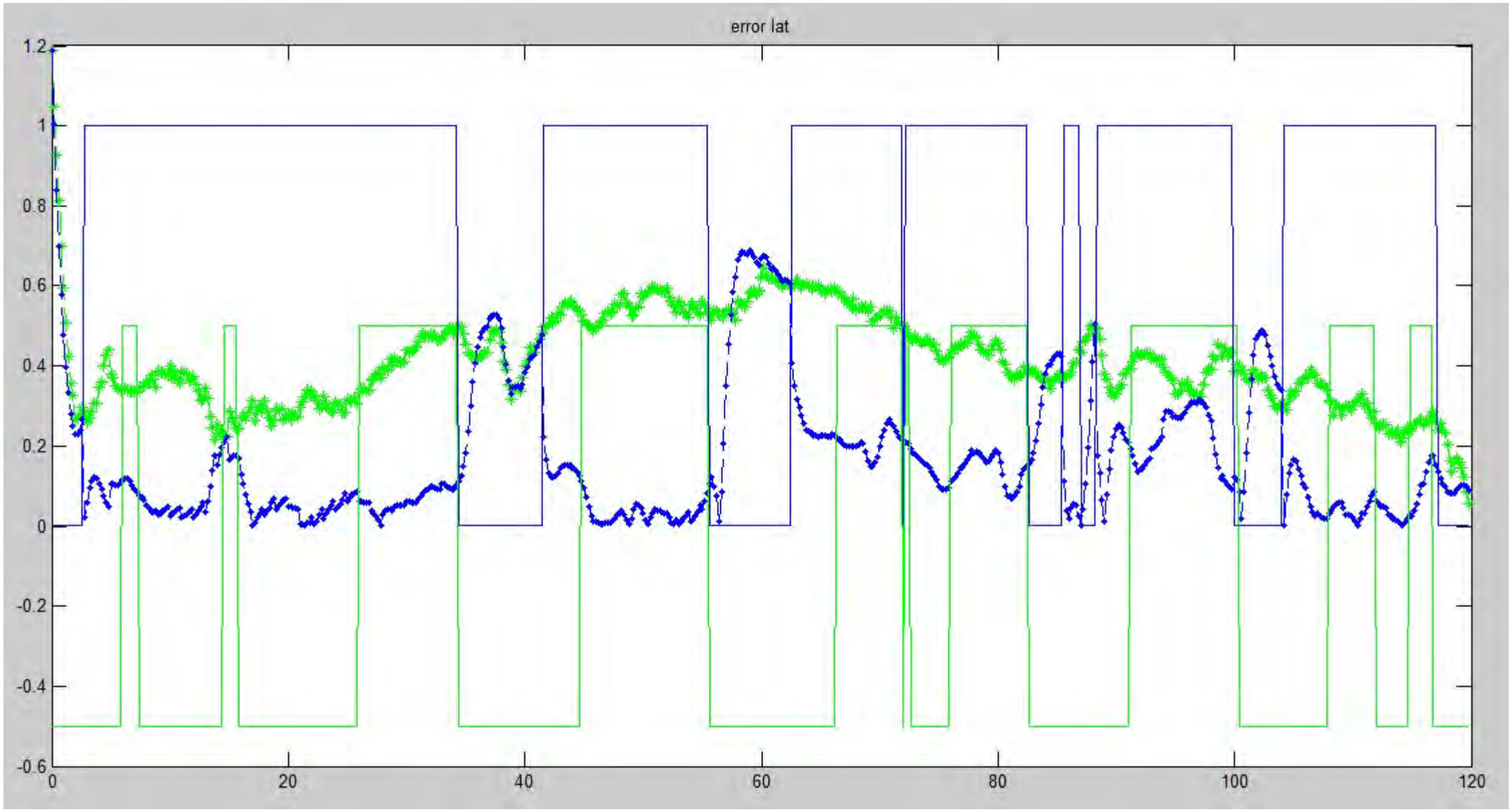


Enhancing maps with lane marking

Modeling



Experimental results



Lateral error

Conclusion

Navigable maps provide pertinent information to assist localization and perception processes

In particular, maps are useful to increase the quality of localization

In terms of

- Precision
- Availability
- Integrity

Associated publications

- Drevelle, V. and Bonnifait, P. “iGPS: Global Positioning in Urban Canyons with Road Surface Maps”, IEEE Intelligent Transportation Systems Magazine, July 2012
- Fouque, C. and Bonnifait, Ph. "Matching Raw GPS Measurements on a Navigable Map Without Computing a Global Position”, IEEE Transactions on Intelligent Transportation Systems, June 2012
- Drevelle, V. and Bonnifait, P. (2011) A set-membership approach for high integrity height-aided satellite positioning. GPS Solutions
- Drevelle, V. and Bonnifait, P. (2011) Global Positioning in Urban Areas with 3-D Maps. 2011 IEEE Intelligent Vehicles Symposium, Baden-Baden
- Drevelle, V. and Bonnifait, P. (2010) Robust Positioning Using Relaxed Constraint-Propagation. IROS 2010, Taipei Taïwan

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session I

Localization & mapping

- **Title: Robot Localization using efficient planar features matching**
  Authors: B. Charmette, E. Royer, Frédéric Chausse and L. Lequievre

- **Title: Application of Visual-Inertial SLAM for 3D Mapping of Underground Environments**
  Authors: A. Ferreira, J. Almeida and E. Silva

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Robot Localization using efficient planar features matching

Baptiste Charmette
baptiste.charmette@univ-bpclermont.fr
Éric Royer, Frédéric Chausse and Laurent Lequievre
Clermont Universités, Institut Pascal, BP 10448, F-63000 CLERMONT-FERRAND
CNRS, UMR 6602, Institut Pascal, F-63177 AUBIÈRE

***Abstract*— Real-time accurate localization is a key component of an autonomous mobile robot. Visual localization algorithms usually rely on feature matching between the current view and a map using point descriptors. Many descriptors such as SIFT or SURF are designed to recognize features seen from different viewpoint. But in a robotic context, the robot movement can be modeled and bring useful information for the matching problem. In this paper we detail a way of matching features with a local 3D model of the features taking advantage of the motion model of the robot. We describe then methods to describe the motion model. The experimental results show how useful the motion model of robot movement is, and prove that use of other sensors can greatly improve precision and robustness of the localization.**

I. INTRODUCTION

Autonomous navigation of a robot along a trajectory is possible provided that its pose can be computed at any time. Indeed, the robot has to stay on the expected trajectory and correct its movement as soon as necessary. When localization relies on vision, it is usually achieved by matching features between the current view and a map of landmarks. In our case, the map is built during an off-line learning step. Then, the robot has to be localized in this map during the autonomous navigation phase. The main difficulty to achieve this goal consists in matching points seen from different viewpoints.

Some feature descriptors such as SIFT [1] or SURF [2] are designed to be almost invariant to changes of viewpoint. But they need a lot of computation time. To achieve real-time performance, many authors [3], [4], [5] have implemented these methods on a Graphical Processing Unit (GPU). Other matching processes [6], [7] use a 3D modeling of the features to make viewpoint-independent descriptor.

However in a robotic context, the robot movement can generally be modeled, because vehicle is moving continuously — without being teleported from one point to an other. It seems interesting to take part of this fact to improve the matching and the localization. For this reason other matching methods [8], [9], [10] based on 3D models of the features have recently been proposed. These methods, instead of generating a viewpoint independent descriptor, consider landmarks as points lying on locally planar surfaces. Then, using an approximate position of the robot, the plane is projected in the current view, making the matching easier. Contrarily to [9], the present work uses monocular vision instead of stereo. And building the map off-line allows to use a much larger map than in the SLAM approach of [8]. These methods can take as much computation time as the previously cited descriptor based method. For this reason, our algorithm is implement on a GPU with CUDA to achieve real-time performance, as described in [11].

The weakest point in this method is that matching part and consequently, pose computation is highly dependent of the predicted position of the robot. In our work we have defined three prediction models and use this algorithm with every one in real conditions. With this experiment we have determined how the prediction can change the result of localization and how localization can be more precise and robust.

After a summary of the algorithm in section II, the prediction model are detailed in part III. After that, our experiment in the whole localization process is described in section IV.

II. LOCALIZATION ALGORITHM

The algorithm is designed to have first a learning stage. During this stage, the vehicle is manually driven in the test area to find some features. These features are considered as planar features, whose orientation and position are computed as described in part II-A. Then in the second stage the vehicle is automatically driven computing its position with the features compared to the images seen by the camera. The localization part is described in part II-B. Result of the localization are combined in the dynamic model of the robot described in part III.

A. Planar feature

Planar feature generation from the learning sequence use the process described in [10]. Some points are tracked on every images, using the Harris [12] interest point detector, and matched according to the Zero Normalized Cross Correlation (ZNCC). Then the structure from motion algorithm [13] computes 3D coordinates of the features and the camera pose of each view.

After that, the features are considered to be lying on a locally planar surface. Considering 2 poses P_i and P_j from where the feature can be seen, an homography $H_{i\to j}$ induced by the plane can be defined to transform the image of the feature seen from P_i into the image seen from P_j. $H_{i\to j}$ depends on the pose P_i and P_j and on the normal to the plane. $H_{i\to j}$ is used to warp the image taken from

P_i in an other view and compared with the real image as shown if the figure 1. If we note I_i^j the image of the

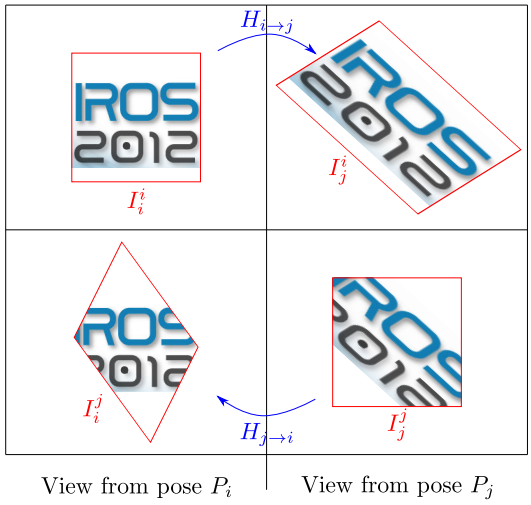


Fig. 1: Summary of the real image and warped image

feature taken from pose P_i and warped by the homography to have the appearance of the feature in the pose P_j. Then for different values of the normal, the image I_i^j is compared with the image I_j^j. An optimization algorithm whose variation parameter is the normal of the surface minimizes the difference between I_i^j and I_j^j to find the value of the normal.

Then a virtual pose P_{ref} is defined to describe the final texture of the patch. For every pose i, the final value of the normal is used to compute $H_{i\to ref}$, homography which transforms image I_i^i in an image I_{ref}^i of the feature as seen from P_{ref}. Then, every image I_{ref}^i is averaged to have the image I_{ref} of the feature. I_{ref} is considered as the appearance of the feature —the texture of the patch— as seen from P_{ref}.

We can note that P_{ref} is not a fronto-parallel pose, but a pose located close to the pose of the image where the feature was seen. This avoids to have an important resampling when the feature was seen from a grazing angle.

After that, the texture I_{ref} is compared to the image I_{ref}^i of the feature to determine the quality of this patch. Indeed, in case of bad matching, or when the feature is not lying on a planar surface, I_{ref} is a very blurred image useless for the localization. Comparison with the image give a quality factor used to remove a feature not usable.

Finally the pose P_i where the feature was observed gives an estimation of the position from where the patch can be seen. The area around these positions is computed to have the observability area. In the localization part, when the camera is not in the observability area, the matching process can be avoided on the feature.

To sum up the patches generated are composed with

- The 3D coordinates of the feature
- The orientation of the normal to the surface
- The planar texture I_{ref}
- The pose P_{ref} associated to the texture
- The observability area

B. Localization

The localization part is done in a second stage, when the vehicle is moving autonomously in the area. The image shot by the camera is compared to the patches and when their positions in the image are found, a triangulation algorithm is applied to compute the camera pose. The chart on figure 2 describe the localization algorithm used to compare features with the image sent by the camera.

1) Pose Prediction: The first step consists in predicting the camera pose P_{pred}, using dynamic model of the robot and the previous position computed. As the robot is moving on the road, the predicted parameters are only its coordinates and the yaw angle. The pitch and roll angle are supposed constant and are not used in the prediction. This prediction is made by the model described in III.

2) Patch Projection: With the pose prediction, the camera position is approximatively known and only patches whose observability area include the camera predicted position are considered. Moreover, as P_{pred} is known with a certain variance, for each patch, a region of interest (*ROI*) is computed defining the area where the patches can be located. The homography induced by the plane is computed and the texture is resampled to obtain image I_{pred} of the patch, as it was observed from the pose P_{pred}. Then a descriptor D_{patch} is computed with the image using for each pixel x the equation 1

$$D_{patch}(x) = \frac{I_{pred}(x) - \overline{I_{pred}}}{\sqrt{\sum_{x\in E_{patch}} (I_{pred}(x) - \overline{I_{pred}})^2 / N_{patch}}} \quad (1)$$

where E_{patch} represent every pixel of I_{pred}, and N_{patch} the number of element of E_{patch}

3) Image processing: At the same time the image from the camera is processed. First, the camera calibration is used to remove any distortion on the image. Then, an interest point detector — the same as the one used for the planar feature generation in section II-A — is applied on the image. After that, for every interest point, a descriptor D_{IP} is computed, using an equation similar to equation 1, using the neighborhood of the point — a window of 16 by 16 pixel in our case — instead of I_{pred}.

4) Matching: Every interest point lying in an uncertainty area of a patch is considered as a candidate to be matched with the patch. To compute their matching score S, the descriptor D_{patch} and the descriptor D_{IP} are combined with the equation 2

$$S = \frac{1}{N} \sum_{x\in E} D_{patch} D_{IP} \quad (2)$$

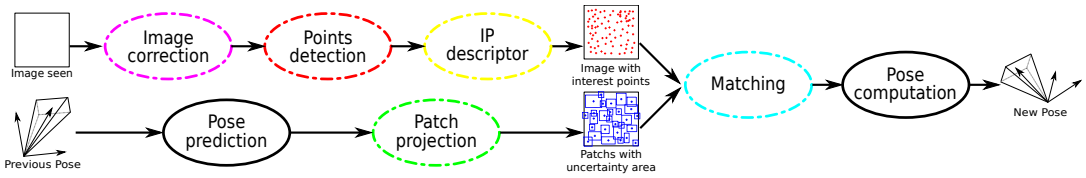


Fig. 2: Chart showing the localization algorithm

where $E = E_{IP} \cap E_{patch}$ is the set of value common with E_{patch} and E_{PI} and N its number of element.

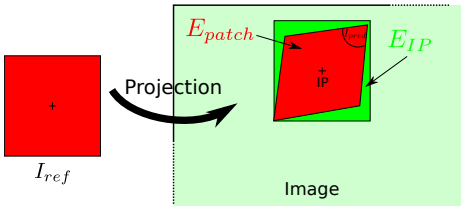


Fig. 3: Patch Projection on an image, where are shown the set E_{PI} neighborhood of an interest point and the set E_{patch}, area where the patch is projected

We can note that, as shown on the figure 3, the reprojection process make the set E_{patch} not fit in the set E_{IP}. For this reason the matching score is not exactly the same as a classic ZNCC score. However, the areas are not very different and the score computed with this method is highly similar to the real correlation computed on the region E.

The matching pairs with a score higher than a threshold — 0.5 in our experiments — are kept. When every candidate matching on every patches have been computed, they are compared and if a patch or IP is present in several different pairs, only the pair with the best score is kept.

After this operation the 2D coordinates of interest point are linked with the 3D coordinates of the patches. With these, the algorithm described in [14] is applied to compute the camera pose and the variance associated.

5) GPU Implementation: The implementation of the algorithm on a classical processor needs a lot of computation time, mainly during the projection of patch. For this reason an implementation on GPU (graphics processing unit) has been done and is described in details in [11]. In particular, the major improvements happen in the resampling part of the algorithm, in the image correction and the patch projection. The resampling is done on the GPU with the texture memory, which is designed to compute hardware interpolation explaining why such an improvement can be done.

An other improvement is due to the massive parallelization of the GPU, which allows to compute several patches simultaneously.

To evaluate the improvement made by the GPU implementation, some measurement of the execution time of the localization have been made. The global time has been decreased to less than 200 ms per image and made it possible to use it in real time for the navigation of a mobile robot. The position is then send to the dynamic model of the robot which will predict the position in the next iteration.

III. Dynamic model of the robot

The position computed with the image analysis is integrated in a dynamic model of the robot. This model is then used to compute the prediction of the next position needed in the first part of the localization algorithm. Three models have been developed for our tests.

A. Simple model

The first model used considers only that the robot is not moving quickly and so uses the previous camera pose as the prediction. With this model, the uncertainty of the prediction is not changing, and the covariance matrix associated to the position is the sum of the covariance matrix of the previous pose and a constant matrix, big enough to insure the new position of the robot is in the uncertainty area. The main advantage of this model is that no assumption are made on the vehicle movement. But the uncertainty area has to be important to consider any change of position. Moreover, the prediction is not very reliable because generally, the vehicle is moving.

B. Constant speed model

The second model uses an extended Kalman filter [15] to predict the evolution of the robot. The model is considered only in 2D, in the ground plane. The main idea of this model is to consider that the linear speed — written $\bar{v}$ — and angular speed — written $\dot{\theta}$ — are constant. The state vector $\underline{X}$ is defined with $\underline{X} = (x, z, \theta, \bar{v}, \dot{\theta})^T$ where θ is the yaw angle. The vehicle is modeled with the tricycle model showed on figure 4. In the discrete time, the value at iteration $k+1$ is defined by equation 3.

$$\begin{pmatrix} x_{k+1} \\ z_{k+1} \\ \theta_{k+1} \\ \bar{v}_{k+1} \\ \dot{\theta}_{k+1} \end{pmatrix} = \begin{pmatrix} x_k - \frac{\bar{v}_k}{\dot{\theta}_k}\left(\cos(\theta_k - \dot{\theta}_k \Delta t_k) - \cos\theta_k\right) \\ z_k - \frac{\bar{v}_k}{\dot{\theta}_k}\left(\sin(\theta_k - \dot{\theta}_k \Delta t_k) - \sin\theta_k\right) \\ \theta_k + \dot{\theta}_k \Delta t_k \\ \bar{v}_k \\ \dot{\theta}_k \end{pmatrix} + \underline{W}_k \quad (3)$$

with Δt_k the elapsed time between iteration k and $k+1$ and $\underline{W}_k$ a noise vector of the same dimension as $\underline{X}$ whose coordinates are supposed uncorrelated zero-mean Gaussian values. In the extended Kalman filter implementation, the covariance matrix Q_k associated to $\underline{W}_k$ is constant. In our

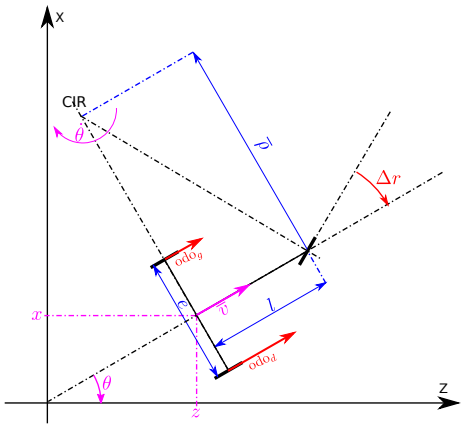


Fig. 4: Model used to describe the vehicle

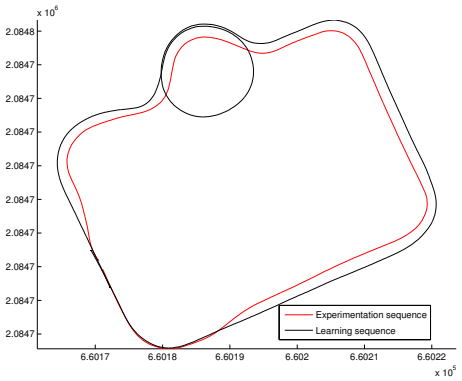


Fig. 5: GPS Track of the learning and test sequence

case, experimental values taken for this matrix is $10^{-5}I$ with I, the identity matrix of size 5.

The vision localization algorithm updates directly the first three values of the state vector using Kalman equation. The noise on the measure is computed in the localization process using the projection error of every features.

This model gives better prediction than the simple model, because vehicle speed is taken in consideration. The prediction errors occur mainly when the vehicle is turning and/or changing its speed.

C. Data fusion with odometry

To help the prediction and predict the change of speed, some information from odometry is used in the last model. The same filter as the constant speed model is used but besides the vision, data from odometry are used to update the filter. The sensors embedded in the vehicle measure the linear speed of the left and right wheel and the angular deviation of the front wheel in the tricycle model. These value are respectively written odo_l, odo_r and Δr and symbolized in red on figure 4. To integrate these values in the filter, the observation function is given by equation 4.

$$\underline{Y}_k = \begin{pmatrix} \text{odo}_l \\ \text{odo}_r \\ \Delta r \end{pmatrix} = \begin{pmatrix} \overline{v} + \dot{\theta} e/2 \\ \overline{v} - \dot{\theta} e/2 \\ \arctan\left(\frac{l\dot{\theta}}{\overline{v}}\right) \end{pmatrix} \tag{4}$$

with l the distance between rear and front wheel and e distance between left and right wheel.

This model can give the best information available at every time, mainly because data from odometry is available at high frequency.

The three models can be used to predict the pose in the localization process. But in the case of autonomous navigation, they can be used at any time — even between image acquisition — to evaluate the current position of the vehicle and, for example send correct orders to the robot actuator.

IV. EXPERIMENT

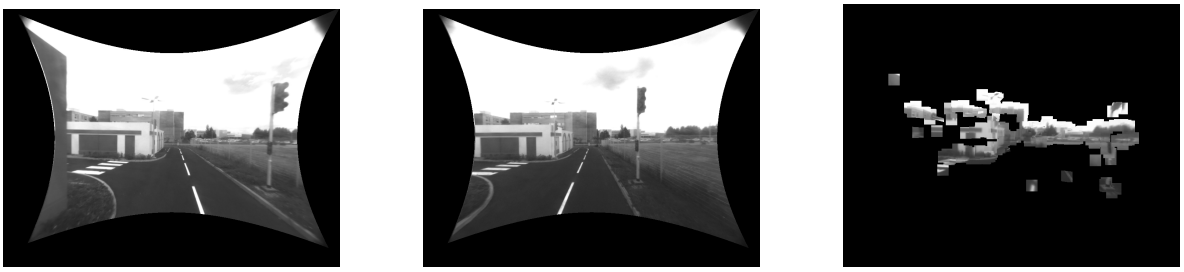
The whole algorithm has been tested to localize a vehicle. First a learning sequence has been acquired with the vehicle moving on the right side of the road. Then the vehicle is manually driven along the left side of the same road, and the algorithm analyzes the camera image to determine the position of the vehicle relatively to the learning sequence. The vehicle has a differential GPS with centimeter precision on board to define a localization reference. Image used have a resolution of 512×384 pixels and the localization was composed of 1232 images. Figure 5 shows the GPS track of the vehicle trajectory while saving learning and localization sequence. Figure 6b shows an image used to generate the patches, 6a shows an image saved in the localization process and 6c the projection of every patch in the pose of this image.

A. Precision Evaluation

The localization process used the patches built in the learning sequence. Obviously, the GPS reference is only used a as the ground truth to compare the result and not used for the localization. To compare the vision results with the GPS localization, the learning trajectory is used to find the transformation needed to convert vision localization in a GPS reference. Then the position found with every prediction model can be compared with the reference.

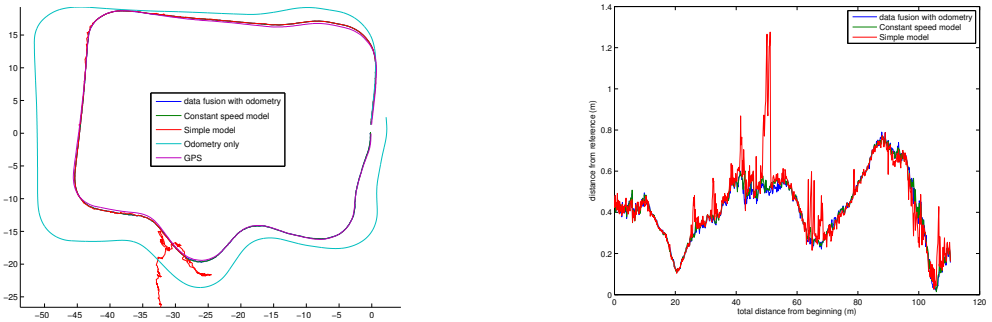
Figure 7 shows the localization results for every method and numerical evaluation of the localization error is summed up in the table I. We can see directly on the top view that with the simple model, localization is failing before the end. The comparison with the reference shown in figure 7b and in the table I are taken on the beginning of the trajectory, when the localization has not failed. Even in this part the first model has less precision than the other. It proves that the simple model gives a quite bad prediction to the algorithm. The fact that localization failed show that prediction is very important in the process. Indeed when the prediction is bad, the patch reprojections are bad and can not be matched with the current view.

For the other localization algorithm, there is no real difference in localization, and no algorithm can be considered more accurate than another. We can conclude that, concern-



(a) Localization image (b) Learning image (c) Patches reconstruction

Fig. 6: Extract from the test Sequence



(a) Top view of the localization

(b) Localization error. Only the part where every method are localized is shown.

Fig. 7: Results of the localization

	average error	standard deviation	median error	maximum error
Simple model	0.441543	0.177010	0.427648	1.276092
Constant speed model	0.422777	0.157316	0.422250	0.767080
Data fusion with odometry	0.419063	0.157680	0.422715	0.789980

TABLE I: Result of the localization process on the beginning of the sequence (where no localization has failed)

ing the precision of the localization, when the prediction is quite correct, it has no influence.

B. Robustness Evaluation

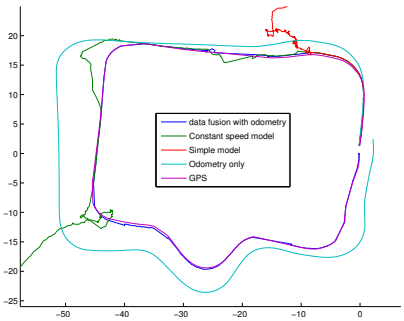
If the precision is the same with or without odometry, the use of an other sensor is not useless. Actually in harder conditions, for exemple if there is some problem with the camera, the use of the odometry improves the localization. To evaluate this point an other experimentation was conducted. The same sequence was used, but several images were removed. The lack of images simulate for example an occultation of the camera or a temporary under or overexposure which is common in the case of indoor-outdoor transition. With this kind of problems, the vehicle can make an important movement between two well exposed images. Results are shown on figure 8. In this experiment, localization using the simple model failed in the first turn with a lack of images. It seems logical because in the case of a turn, the features are moving a lot in the image. If the prediction is still the same pose as before, no projected patch can be matched on the current view.

Using the model without odometry creates several differences and localization failed in the middle of the trajectory. Figure 8b shows that even in the first part, localization is quite bad after every lack of image, mainly when the vehicule is turning. It is confirmed with statiscal analysis of the data in the table II. The use of odometry greatly improves the prediction, having a correct localization in every situation.

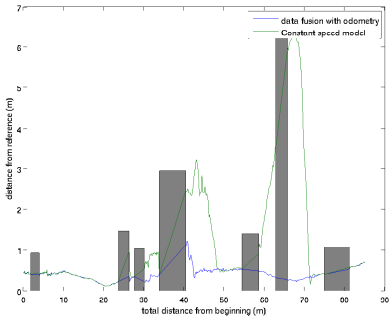
V. CONCLUSIONS AND FUTURE WORKS

A. Future Works

To use directly Kalman filter equation in the `Constant speed model`, the covariance on the model and on the measured value have to be known. Measure variance is evaluated with the reprojection error given by the pose computation. The model uncertainty is not so easy to know. In fact, it symbolizes the time variation of the speed, or in other word the acceleration. For the moment, the uncertainity is considered as constant and applied in the same way on every value. A future work could be to model the noise only as an acceleration part on every iteration and combine it in the Kalman filter. This can decrease uncertainty on the prediction method and then decrease computation time of



(a) Top view of the localization



(b) Localization error. Only the part where both methods are localized is shown. Gray rectangles are the parts where no images are available

Fig. 8: Results of the localization after removing images

	average error	standard deviation	median error	maximum error
Constant speed model	1.13	1.42	0.47	6.43
Odometry fusion	0.41	0.15	0.41	1.23

TABLE II: Result of the localization process on the beginning of the sequence (where no localization has failed) where several images have been removed

the localization because less matching candidates would be analysed.

An other way of improvement could be to use the prediction to make a previous selection of the patches before the projection. Although the observability area is filtering several patches, a lot of them are still processed and are not useful, because they are not matching any interest point. A previous analysis could avoid to make too many projection, and prefer to project only the best patches which are sufficient to achieve a good localization.

B. Conclusions

We have shown that a new matching method based on planar feature modelling and reprojection can be used for the robust pose computation of a mobile robot. Thanks to an efficient GPU implementation, the localization algorithm can be used for real time autonomous navigation. The experimental resuts outline that the prediction step is really important for the robustness of the algorithm. the information given by other sensor such as odometry can also improve the algorithm particularly in difficult cases when the position computed with vision momentarily unavailable.

VI. ACKNOWLEDGMENTS

This work was supported by the french ANR national agency in the CityVIP project

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, 1999, pp. 1150–1157.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] S. Sinha, J. Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching," in *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, vol. 278. Citeseer, 2006.

[4] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008*, 2008, pp. 1–8.

[5] M. Schweitzer and H.-J. Wuensche, "Efficient Keypoint Matching for Robot Vision using GPUs," in *Fifth IEEE Workshop on Embedded Computer Vision (ECV'09)*, 2009.

[6] K. Koser and R. Koch, "Perspectively invariant normal features," in *ICCV 2007*, 2007, pp. 1–8.

[7] C. Wu, B. Clipp, X. Li, J. Frahm, and M. Pollefeys, "3D model matching with Viewpoint-Invariant Patches (VIP)," in *CVPR 2008*, 2008, pp. 1–8.

[8] N. Molton, A. Davison, and I. Reid, "Locally planar patch features for real-time structure from motion," in *BMVC*, 2004.

[9] C. Berger and S. Lacroix, "Using planar facets for stereovision SLAM," in *IROS*, 2008, pp. 1606–1611.

[10] B. Charmette, É. Royer, and F. Chausse, "Matching Planar Features for Robot Localization," in *International Symposium on Visual Computing*. Springer, 2009, pp. 201–210.

[11] B. Charmette, F. Chausse, and É. Royer, "Efficient planar features matching for robot localization using gpu," in *Sixth Workshop on Embedded Computer Vision held in conjuction with CVPR 2010*, Jun. 2010.

[12] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, 1988.

[13] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, pp. 237–260, 2007.

[14] H. Araujo, R. Carceroni, and C. Brown, "A fully projective formulation to improve the accuracy of Lowe's pose-estimation algorithm," *Computer vision and image understanding(Print)*, vol. 70, no. 2, pp. 227–238, 1998.

[15] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

Application of Visual-Inertial SLAM for 3D Mapping of Underground Environments

António Ferreira, José Almeida and Eduardo Silva
INESC TEC - INESC Technology and Science
(formerly INESC Porto) and ISEP/IPP - School
of Engineering, Polytechnic Institute of Porto

***Abstract*—The underground scenarios are one of the most challenging environments for accurate and precise 3d mapping where hostile conditions like absence of Global Positioning Systems, extreme lighting variations and geometrically smooth surfaces may be expected. So far, the state-of-the-art methods in underground modelling remain restricted to environments in which pronounced geometric features are abundant. This limitation is a consequence of the scan matching algorithms used to solve the localization and registration problems.**

This paper contributes to the expansion of the modelling capabilities to structures characterized by uniform geometry and smooth surfaces, as is the case of road and train tunnels. To achieve that, we combine some state of the art techniques from mobile robotics, and propose a method for 6DOF platform positioning in such scenarios, that is latter used for the environment modelling.

A visual monocular Simultaneous Localization and Mapping (MonoSLAM) approach based on the Extended Kalman Filter (EKF), complemented by the introduction of inertial measurements in the prediction step, allows our system to localize himself over long distances, using exclusively sensors carried on board a mobile platform. By feeding the Extended Kalman Filter with inertial data we were able to overcome the major problem related with MonoSLAM implementations, known as scale factor ambiguity. Despite extreme lighting variations, reliable visual features were extracted through the SIFT algorithm, and inserted directly in the EKF mechanism according to the Inverse Depth Parametrization. Through the 1-Point RANSAC (Random Sample Consensus) wrong frame-to-frame feature matches were rejected.

The developed method was tested based on a dataset acquired inside a road tunnel and the navigation results compared with a ground truth obtained by post-processing a high grade Inertial Navigation System and L1/L2 RTK-GPS measurements acquired outside the tunnel. Results from the localization strategy are presented and analyzed.

I. Introduction

Over the last few years some successful underground mobile modelling implementations were documented [1] [2] [3]. These approaches, designed specifically to operate in mines, are characterized by one common aspect: they all use laser range finder sensors as the main (and in some cases the only) source of information. The model is built by placing laser range finder scans in a virtual three-dimensional world – process called registration. For this purpose, relative position and orientation between scans have to be determined. In previous approaches, this task is accomplished via a scan matching algorithm [7], which restricts the systems to non-uniform structures, since this technique requires that notorious and well-differentiated geometric features stand out along overlapping scans.

Our work extends the underground mobile modelling systems to galleries characterized by uniform and smooth surfaces. In this type of scenario the scan matching approaches are condemned to failure, so the previous state-of-the-art systems become ineffective. Without artificial landmarks and no access to Global Positioning Systems, self-localization becomes an hard problem. In inertial based localization the errors accumulated over time cause a monotonic growth in localization uncertainty. On the other hand, a vision based approach may be affected by the lighting conditions, additionally, the parametrization of landmarks far from the cameras raises extra difficulties due to the depth uncertainty.

Similarly to [3], our solution uses 2D laser range finders to gather a sequence of vertical scans along the gallery. Absolute position and orientation of each scan is computed by an independent localization process, that estimates the systems' trajectory based on inertial measurements and a sequence of images.

We employ an alternative localization solution to overcome both the structural monotony and the lack of Global Positioning Systems, adopting the SLAM (Simultaneous Localization and Mapping) concept [8] [9] to estimate the platforms localization in 6DoF (Six Degrees of Freedom). Following the traditional approach, the probabilistic SLAM algorithm is based on the EKF (Extended Kalman Filter). Since for landmarks far from the cameras, stereoscopic systems do not provide satisfactory depth measurements, a visual monocular algorithm was implemented instead, ensuring tracking of landmarks at any depth.

In order to identify visual landmarks to be used in the SLAM algorithm, highly distinctive visual features, invariant to scale, rotation and linear illumination variations, are extracted from the images using the SIFT algorithm [11]. To each feature is assigned at least one descriptor, that embodies the image properties in the features' neighborhood. The descriptors are used to establish the frame-to-frame feature matches.

Our system combines another advanced state-of-the-art methods such as Inverse Depth Parametrization [5], and the 1-Point RANSAC algorithm [6], for outlier rejection.

Through the Inverse Depth Parametrization, undelayed initialization of landmarks within the EKF framework be-

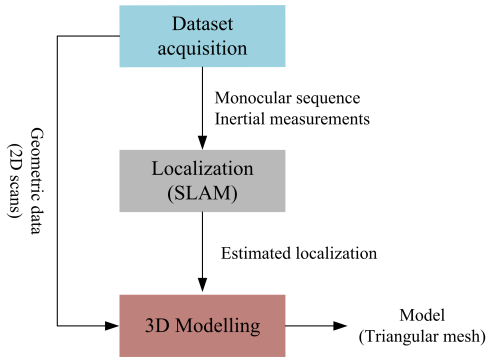


Fig. 1: High level system architecture

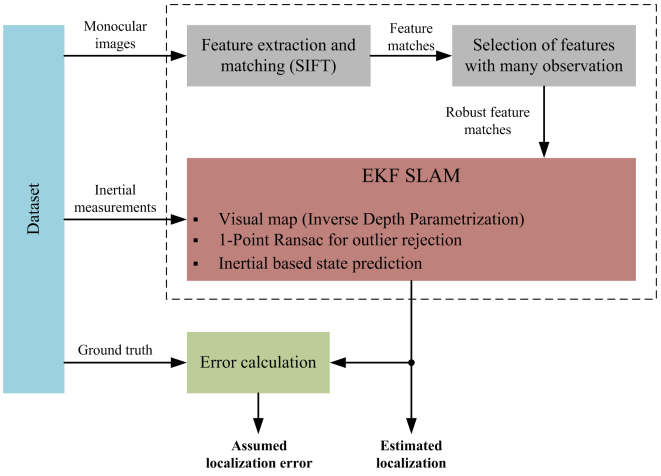


Fig. 2: Localization algorithm overview

comes possible. However another major problem of monocular SLAM applications still needs to be solved: a single camera moving through the scene does not provide metric measurements, leading to scale ambiguity in the estimated map and motion. As suggested in [4] inertial measurements, provided by a low-cost IMU, feed the filter with metric data in order to prevent the scale factor degeneration. This strategy keeps the map and motion estimates constrained to the meaningful metric system, in our case for distances over more than one hundred meters.

To build the model, all vertical cross sections are placed on a common reference frame according to the localization estimates, resulting in a point cloud model, which is finally converted into a triangular mesh through the Ball Pivoting Algorithm [10], to reach a more explicit representation without information losses. Texture captured by the cameras is also added to the model to enhance the visual realism.

This document is organized as follows: Section II presents a brief architecture description with emphasis on the localization and modelling algorithms. Section III is devoted to the dataset acquisition that takes places inside a road tunnel. We then present and discuss our implementation results (Section IV) and finally, Section V, provides a conclusion and sets some future goals.

II. System Architecture

Our system is divided in three main blocks, executed by the following order: data acquisition, localization and three-dimensional modelling (see Fig. 1).

In the first step, a sensor platform mounted on board a car is used to collect a wide range of synchronized measurements inside the underground galleries, including images captured by two CCD cameras, 2D scans from two laser range finders and inertial measurements provided by a low cost inertial measurement unit. The platform carries also a INS/GPS system that gives accurate ground truth information, used to measure the performance of our localization strategy.

The localization estimation and modelling tasks are preformed offline based on this data, according to the methods described next.

A. Localization Algorithm

In underground galleries it is expected to find reliable visual features that can be used as reference points to build the SLAM map. The process starts with a feature pre-selection stage (see Fig. 2) to fulfill the following objectives:

- Reduce the computational complexity of the SLAM cycle, by performing feature extraction and frame-to-frame matching in advance. The feature extraction is accomplished by the SIFT algorithm [11], that produces descriptors invariant to scale, orientation, and linear illumination changes, used to compute the frame-to-frame feature matches;
- Identify features with large number of observations and use only those to build the map. By doing so, we intend to minimize the computational demands, ensuring that all landmarks in the map persist over an acceptable frame interval.

1) State Vector: The SLAM cycle is implemented according to the EKF method. The state vector stores the localization and map states. Since the system does not have prior information about the environment, the initial state vector includes only 9 states related to the platforms' localization: position x^n, orientation Θ^n (expressed in terms of Euler angles) and velocity v^n, all defined in the local level reference frame (see Fig. 3).

$$x(k) = (x_b)^n(k) = \begin{bmatrix} x^n(k) \\ \Theta^n(k) \\ v^n(k) \end{bmatrix} \quad (1)$$

As new landmarks are observed, the state vector is expanded to accommodate the respective states (equation 2).

$$x(k) = \begin{bmatrix} (x_b)^n(k) \\ L_1(k) \\ L_2(k) \\ \vdots \\ L_n(k) \end{bmatrix} \quad (2)$$

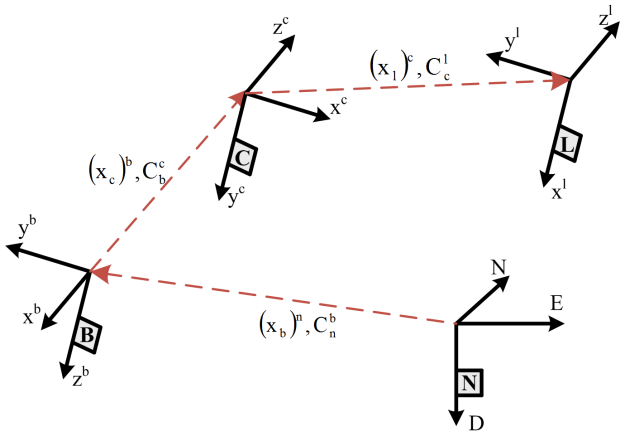


Fig. 3: Reference frames used in the localization and modelling algorithms. Local level frame (N), body frame (B), camera frame (C) and laser range finder frame (L).

Initially, each landmark L_i is coded in the SLAM map using the Inverse Depth Parametrization [5], which requires six parameters (Fig. 4): position of the cameras' optical center at the moment of first observation $[x_i^n\ y_i^n\ z_i^n]$, azimuth θ_i and elevation ϕ_i angles of the projection ray that passes through the optical center and the landmark, and finally the inverse of the distance ρ_i between the optical center and the landmark in the world (inverse depth).

$$L_i = [x_i^n,\ y_i^n,\ z_i^n,\ \theta_i,\ \phi_i,\ \rho_i]^T \tag{3}$$

The state uncertainty of this overparameterized representation can be modelled by Gaussian distributions, regardless to the distance between the landmark and the camera, therefore this is an efficient and accurate solution for undelayed initialization of new landmarks within the EKF. The EKF computational complexity grows quadratically with respect to the state vector dimension, so when the uncertainty in the landmark's location reveals a Gaussian behavior, indicated by the linearity index introduced in [12], the conversion to the standard Cartesian representation is accomplished applying the formula below:

$$\begin{bmatrix} L_{xi} \\ L_{yi} \\ L_{zi} \end{bmatrix} = \begin{bmatrix} x_i^n \\ y_i^n \\ z_i^n \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) \tag{4}$$

being $[L_{xi},\ L_{yi},\ L_{zi}]$ the Cartesian coordinates of the landmark and $m(\theta_i, \phi_i)$ a unitary vector (see Fig. 4), calculated from the azimuth and elevation angles:

$$m(\theta_i, \phi_i) = \begin{bmatrix} -cos(\phi_i)sin(\theta_i) \\ sin(\phi_i) \\ cos(\phi_i)cos(\theta_i) \end{bmatrix} \tag{5}$$

2) Landmark Initialization: From the six parameters that define an Inverse Depth landmark, only the azimuth and elevation angles need to be computed, since the camera position is already defined in the state vector, and the initial inverse depth consists on a fixed value defined in advance. To compute the angles, the feature is first projected from the image to the camera reference frame, using the pinhole camera model. A distortion model is applied next to compensate for the lens distortion. From this operation results a three-dimensional non-unitary vector h^c with the same orientation as the projection ray. The vector expressed in the navigation frame is given by:

$$h^n = C_b^n C_c^b h^c \tag{6}$$

where C_b^n and C_c^b are the rotations matrices from the body frame to the navigation frame and from the camera frame to the body frame, respectively (see Fig. 3).

From h^n, the orientation angles can be finally computed as follows:

$$\begin{bmatrix} \theta_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} arctan(-h_x^n,\ h_z^n) \\ arctan\left(h_y^n,\ \sqrt{(h_x^n)^2 + (h_z^n)^2}\right) \end{bmatrix} \tag{7}$$

3) Landmark Prediction and Outliers Rejection: At the update step of the Extended Kalman Filter the position of the features observed in the image is compared to the expected projection of the map landmarks in the image. The projection of a landmark in the map to the image starts with the transformation from the navigation frame to the camera frame:

$$h^c = C_b^c C_n^b \left(\rho_i \left(\begin{bmatrix} x_i^n \\ y_i^n \\ z_i^n \end{bmatrix} - (x_b)^n - C_b^n (x_c)^b \right) + m(\theta_i, \phi_i) \right) \tag{8}$$

The distortion model is then applied to h^c, followed by the pinhole model, to determine the projection in the image.

Finally, wrong feature matches are rejected through the 1-Point RANSAC algorithm [6], that takes into account the prior probabilistic distributions maintained by the EKF to reduced the minimal sample size to only one feature match, significantly reducing the computational complexity associated with the standard RANSAC algorithm.

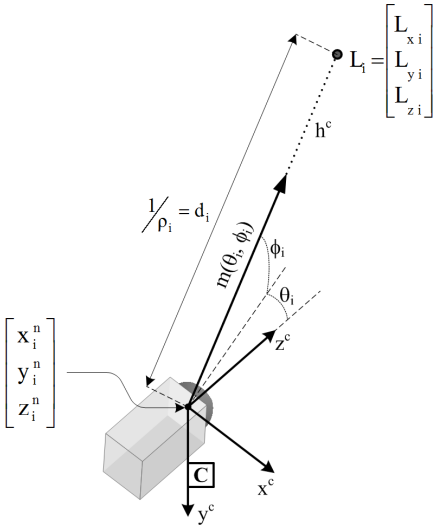


Fig. 4: Representation of the Inverse Depth parameters

4) Inertial Based State Prediction: To avoid the scale factor ambiguity, the main limitation of monocular SLAM caused by the absence of metric information, inertial measurements from a low cost IMU are injected in the EKF prediction step. Since the map landmarks are static, only the platform localization states are subjected to the motion model, that consists on the inertial mechanization in the local level reference frame, respecting the following equations:

$$\begin{bmatrix} x^n(k) \\ \Theta^n(k) \\ v^n(k) \end{bmatrix} = \begin{bmatrix} x^n(k-1) + v^n(k)\Delta t \\ \Theta^n(k-1) + E_b^n w^b(k)\Delta t \\ v^n(k-1) + \left(C_b^n a^b(k) + g^n\right)\Delta t \end{bmatrix} \quad (9)$$

where the IMU inputs are identified by a^b and w^b, respectively the linear accelerations and angular velocities, measured in the body reference frame. C_b^n is the direction cosine matrix obtained from the platform orientation and E_b^n is a 3 by 3 matrix that converts the angular velocities into the Euler angles rate of change:

$$E_b^n = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \quad (10)$$

B. Modelling Algorithm

The three-dimensional model is constructed by placing all gallery cross-sections, taken by the vertical laser range finder, into a common coordinate system.

First, laser range finder scans, initially expressed in polar coordinates, are converted to the Cartesian coordinate system with origin matching the center of the laser range finder. Next, specific position and orientation of each scan is derived from the two closest localization points in time. Given the calibration parameters that describe the spatial relationship between sensors, determined in advance, and using the calculated scan localization, all vertical cross-sections are transformed to the local level frame according to the formula below:

$$P^n = C_b^n \left(C_c^b \left(C_l^c \left(P^l - (x_l)^c \right) - (x_c)^b \right) - (x_b)^n \right) \quad (11)$$

where P^n is the final point in the local level frame, whereas P^l refers to the original point in the sensor Cartesian system. The rotation matrices C_c^b and C_l^c establish the rotation from camera to body and laser to camera reference frames, respectively (see Fig. 3). Whereas $(x_c)^b$ define the camera position in the body frame and $(x_l)^c$ the laser position with respect to the camera frame. Finally C_b^n and $(x_b)^n$ enclose the rigid body transformation from the body to the local level reference frame.

After applying formula (11) to all points of all scans, a point cloud model is achieved. Usually, the interpretation of point clouds is not easy due to lack of surfaces. To improve the scene's perception, original surfaces are reconstructed by converting the point cloud into a triangular mesh, using the Ball Pivoting Algorithm (BPA) [10]. The models realism is also enhanced by introducing texture information captured by the cameras.

To reduce the noise and produce smoother surfaces, a Laplacian filter is applied to the whole triangular mesh, computing a new position for each vertex according to local information given by adjacent points.

Both the point cloud model and the triangular mesh are coded in the VRML format to be displayed in a virtual reality application.

III. Dataset Acquisition

Solving the localization and modelling problems demands previous acquisition of a variety of measurements. To this purpose different types of sensors where assembled in a rigid platform (see Fig. 5), which in turn is mounted on top of a car.

The vertical cross-sections are taken by the vertical laser range finder (SICK LMS-200) at 75Hz with an angular resolution of 1°. There are two pointing-forward cameras (JAI CB-080GE), arranged in a stereoscopic configuration, with a resolution of 1032(h)x778(v) and controlled by an external trigger at a frame rate of 7 fps. Only the images from the left camera are used in our SLAM system.

The low cost IMU (MicroStrain 3DM-GX1), placed above the left camera, gives the linear acceleration and angular velocity measurements used in the EKF prediction step, at a frequency of 100Hz.

Ground truth with a 400 Hz rate is obtained by a tactical grade INS/GPS system (iMAR iNAV-FMS-E) placed in the center of the platform. This system provides raw inertial data and GPS measurements acquired outside the gallery, that are post-processed in a commercial software (Waypoint Inertial Explorer) to produce an accurate trajectory estimation. This trajectory is only used as ground truth to evaluate the SLAM performance.

All system reference clocks are synchronized with respect to GPS clock, to assure a consistent time base.

The data acquisition experiment took place on a road tunnel with approximately 140 meters located at Vilar de Luz – Porto (see Fig. 6). All data were correctly logged. However the

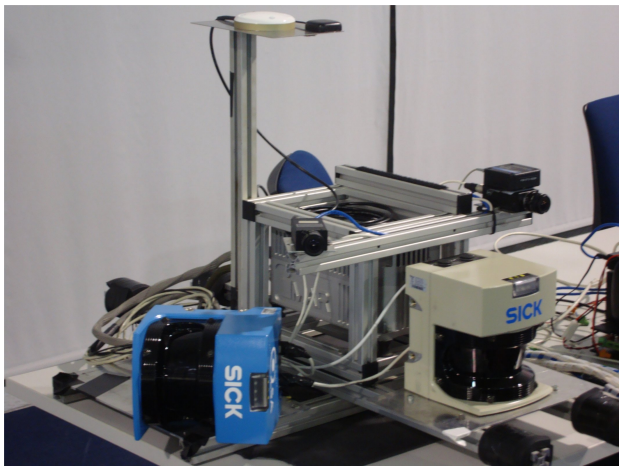


Fig. 5: Sensor platform used for data acquisition

Fig. 6: Preparation for the data acquisition experiment in the tunnel area

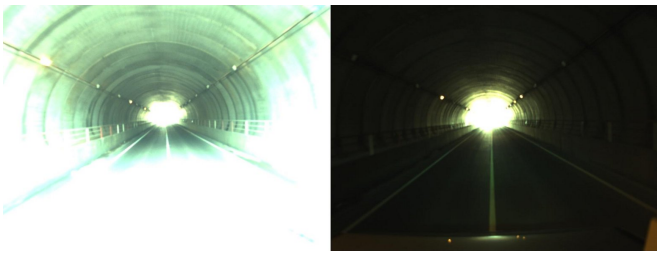


Fig. 7: Image instability as consequence of the illumination variations along the tunnel.

images reflect the huge lighting variations between the interior and exterior of the tunnel (see Fig. 7).

IV. Results

An accurate localization estimate is crucial to obtain a reliable model reproducing the real gallery characteristics. Using the ground truth trajectory the error associated with the estimated localization is determined. Furthermore, to realize the benefits of fusing inertial and visual measurements, both inertial navigation and MonoSLAM approaches were implemented, and the results are compared with the ones achieved by the inertial and visual SLAM approach.

The errors in the position states for each method are outlined in Fig. 8. The path calculated by MonoSLAM shows the worst results due to the scale ambiguity, accumulating an error of 11.7 meters. As expected, inertial navigation drifts with time due to error integration, resulting in a total drift of 8.7 meters. Our approach produces the smallest error, showing the advantage of inertial and visual data fusion, with a maximum value of 1.29 meters and an error of 0.95 meters at the final position. The insertion of inertial measurements in the MonoSLAM mechanism successfully prevents the scale factor ambiguity, whereas visual data contributes to the inertial drift compensation, particularly to the orientation states correction.

The distribution of the features along the image is shown in Fig. 9. The image space is equally divided in four quadrants and an histogram is computed for each portion. It can be seen that the top quadrants provide the most reliable features, in terms of number and long observation sequence. The landmarks introduced in the SLAM map are observed over more

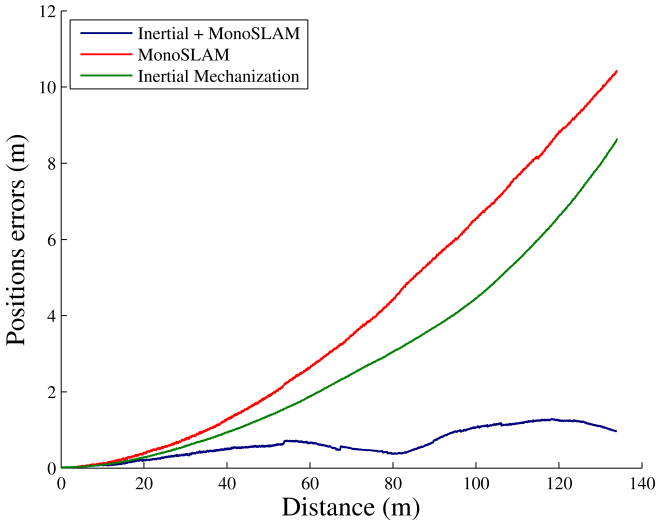


Fig. 8: Position errors produced by each localization strategy: SLAM fusing inertial and visual data (blue line), inertial mechanization (green line) and monocular SLAM (red line)

then 6 frames, and the most persistent ones last a maximum of 50 frames. The short periods of observation reduce the possibility of observing sufficient parallax to convert inverse depth landmarks to the Cartesian form (see Fig. 10).

As the system approaches the end of the tunnel, the effects of image saturation are visible at the final moments in Fig. 10. In the last 20 meters, the 1-Point RANSAC algorithm rejects a considerable amount of wrong feature matches, and the respective landmarks are deleted from the SLAM map.

As previously mentioned, the point cloud models can become really hard to interpret, depending on the view point

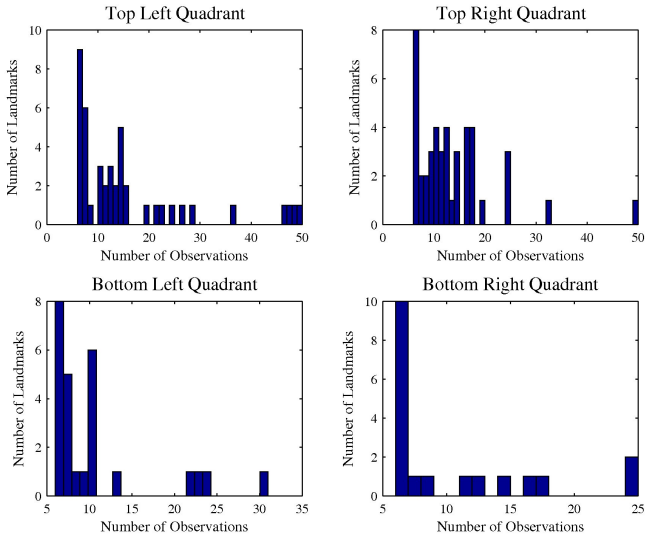


Fig. 9: Histograms that represent the total number of features in respect to the number of observations, for image sub-regions of equal size.

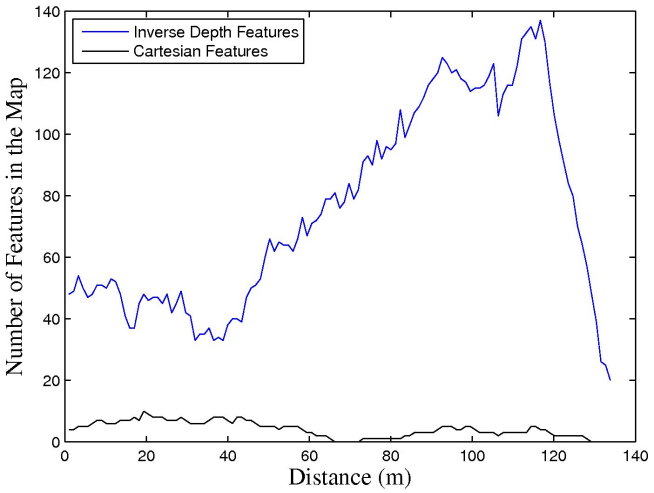


Fig. 10: Landmarks parametrization in the SLAM map along the trajectory.

and scale. In order to reach a more explicit and realistic representation, a triangular mesh is constructed from the point cloud without data losses, through the Ball Pivoting Algorithm. In the final step the surfaces are filtered by a Laplacian smoother, and texture acquired by the cameras is added to the model (see Fig. 11).

V. Conclusion

The development of a mobile modelling system for large scale underground environments raises some difficult challenges, especially when dealing with monotonous geometry. Based on inertial and visual data we have implemented a localization method that does not depend on the geometric properties of the environment, thus it is specifically suited to operate inside smooth shape galleries like traffic tunnels.

Through localization results the benefit of fusing inertial data within the MonoSLAM strategy became evident. In the most aggressive configuration, with a pointing forward camera, forward motion and large illumination variance, our localization estimate reached an error of 0.95% of the total displacement, which constitutes a quite impressive accomplishment given the low cost sensors used.

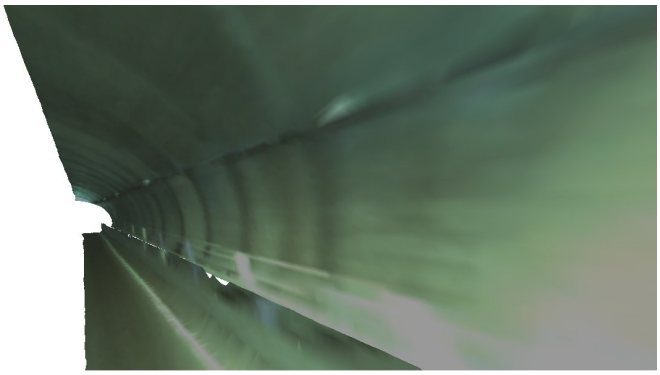


Fig. 11: Triangular mesh model after Laplacian filtering.

Despite the poor image quality, reliable visual features and descriptors where extracted by the SIFT algorithm, exploiting the algorithm's immunity to rotation scale and linear illumination variations, enabling robust frame-to-frame feature matching.

In the future, localization accuracy could be improved by adding other types of information, for instance, laser range finder measurements to provide a better approximation of the landmarks initial depth. A stereo vision system will also be implemented to enable instant computation of close landmark coordinates. The use of cameras with larger field of view will also be beneficial, enabling the observation of landmarks with high parallax and hence low depth uncertainty.

Acknowledgment

This work was supported by QREN – Project n° 7865 "FlexiMap3D".

References

[1] Andreas Nüchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg and Sebastian Thrun, *6D SLAM with an Application in Autonomous Mine Mapping*, IEEE International Conference on Robotics and Automation (ICRA), pages 1998–2003, 2004.

[2] Daniel Huber and Nicolas Vandapel, *Automatic Three-dimensional Underground Mine Mapping*, The International Journal of Robotics Research, volume 25, pages 7–17, January 2006.

[3] Sebastian Thrun, Dirk Hähnel, David Ferguson, Michael Montemerlo, Rudolph Triebel, Wolfram Burgard, Christopher Baker, Zachary Omohundro, Scott Thayer and William Whittaker, *A System for Volumetric Robotic Mapping of Abandoned Mines*.

[4] Pedro Pinies, Todd Lupton, Salah Sukkarieh, Juan D. Tardós, *Inertial Aiding of Inverse Depth SLAM Using a Monocular Camera*, IEEE International Conference on Robotics and Automation (ICRA), pages 2797–2802, 2007.

[5] Javier Civera, Andrew J. Davison and J. M. M. Montiel, *Inverse Depth Parametrization for Monocular SLAM*, IEEE Transactions on Robotics, volume 24, number 5, pages 932–945, October 2008.

[6] Javier Civera, O. Garcia, Andrew J. Davison and J. M. M. Montiel, *1-Point RANSAC for EKF-Based Structure from Motion*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2009.

[7] Paul J. Besl and Neil D. McKay, *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), pages 239–256, 1992.

[8] R. Smith, M. Self and P. Cheeseman, Estimating Uncertain Spatial Relationships in Robotics, In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, Springer-Verlab, 1990.

[9] J. J. Leonard and Durrant H. Whyte, *Simultaneous Map Building and Localization for an Autonomous Mobile Robot*, IEEE International Conference on Intelligent Robots and Systems (IROS), Osaka, Japan, 1991.

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, *The Ball-Pivoting Algorithm for Surface Reconstruction*, IEEE Transactions on Visualization and Computer Graphics (TVCG), 5(4), pages 349–359, 1999.

[11] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision (IJCV), 60(2), pages 91–110, 2004.

[12] Javier Civera, Andrew J. Davison, J. M. M. Montiel, *Inverse Depth to Depth Conversion for Monocular SLAM*, International Conference on Robotics and Automation (ICRA), pages 2778–2783, 2007.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session II

Multiple Vehicles/Robots & Interaction

- **Keynote speaker: Alberto Broggi (Parma University, Parma, Italy)**
  Title: Thoughts on perception for intelligent vehicles
  Co-Authors: P. Grisleri and P. Zani

- **Title: Multiple Robots in a Cooperating Task: Exploration and Mapping**
  Authors: A.M. Neto, P. Rosa, T.E. Alves de Oliveira and P.C. Pellanda

- **Title: Dynamic Obstacle Avoidance Strategies using Limit Cycle for the Navigation of Multi-Robot System**
  Authors: A. Benzerrouk, L. Adouane and P. Martinet

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session II

Keynote speaker: **Alberto Broggi**
(Parma University, Parma, Italy)

Thoughts on perception for intelligent vehicles

Co-Authors: P. Grisleri and P. Zani

Abstract : Many successful implementations of intelligent vehicles are using laser based technology to obtain a 360 degrees view of the area surrounding the vehicle. Such technology is able to provide very dense 3D point clouds covering an extended range. The talk will compare this technology to other lower cost alternatives, such as vision, and discuss some possible implementations.

Biography: Dr. Alberto Broggi is a professor of Computer Engineering at the University of Parma in Italy, and CEO of the VisLab spinoff company. As a pioneer of machine vision applied to driverless cars and unmanned vehicles, he led his groups' efforts in many projects involving autonomous vehicles. Some important milestones include the ARGO project and its test in 1998, the TerraMax entry in the DARPA Challenges from 2004 to 2007, and the BRAiVE project. Under his leadership VisLab organized the first intercontinental driverless trip in history, named VIAC - VisLab Intercontinental Autonomous Challenge: a 13,000 km driverless trip from Italy to China in 2010.
He acted as Editor-in-Chief of the IEEE Trans on Intelligent Transportation Systems from 2004 to 2008. For the term 2010-2011 he served the IEEE Intelligent Transportation Systems Society as President.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Thoughts on perception for intelligent vehicles

Alberto Broggi, Paolo Grisleri and Paolo Zani
VisLab – Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma, ITALY
http://www.vislab.it
{broggi,grisleri,zani}@vislab.it

***Abstract*— Many successful implementations of intelligent vehicles are using laser based technology to obtain a 360 degrees view of the area around the vehicle, providing a very dense 3D point cloud that covers an extended range. However, such technology is still too expensive to be a candidate for series production, and its integration requirements are hardly compatible with cost and style constraints dictated by the mainstream automotive market. On the other hand computer vision is reaching close enough results in terms of sensing performance to be a viable alternative. Vision also brings additional advantages such as a much lower price and less integration requirements. This paper compares the two technologies, and discusses some possible implementations.**

I. INTRODUCTION

Driving in urban traffic needs 360 degrees perception capabilities in a variety of conditions, in order to timely react to a constantly changing scenario. This requirement holds for humans as well as for intelligent and autonomous vehicles, but while in the first case the sensing system is fixed, in the latter sensors selection becomes a key design decision, which must satisfy a number of constraints besides environment mapping performance.

During the 2007 DARPA Urban Challenge two fundamentally different approaches emerged to achieve the required long-range, fast and accurate 3D reconstruction of the vehicle surroundings: one exploiting high-end LIDAR units [1], [2], [3], the other heavily relying on computer vision [4]. Since then this trend has continued, with more prototypes being developed around the world: some notable examples include Google's driverless car and VisLab's BRAiVE autonomous vehicle [5], both depicted in Fig.1.

Other technologies, such as RADAR and sonar sensors, have been widely tested as well, and are currently used in a number of commercial advanced driving assistance systems [8]; however, usually they do not provide a representation of the vehicle surroundings detailed and accurate enough to perform autonomous navigation, and as such will not be discussed in the following. Instead, this work will focus on the key differences between the two most promising technologies to date, pointing out their key strengths and weaknesses.

As it will be shown in the rest of this paper, both LIDAR and computer vision can provide a comparable amount of information, but with very different trade-offs in terms of cost, integration, field of view, and failure modes, all aspects which have a direct impact on the applicability of a given technology in the mainstream automotive market.

(a)

(b)

Fig. 1. Autonomous vehicle designs compared: a) Google's driverless car, featuring a prominent high-end LIDAR unit as its main sensor, and b) VisLab's BRAiVE more integrated design, heavily relying on computer vision: 10 cameras have been mounted on-board, following the experience gained while developing the perception layer for the TerraMax autonomous vehicle during the 2005 and 2007 DARPA Challenges [6], [7].

II. ENVIRONMENT MAPPING TECHNOLOGIES COMPARED

Both LIDAR units and stereo cameras provide data as a 3D point cloud, so the most straightforward way of comparing the mapping capabilities of the two sensors is to analyze the characteristics of the information they produce in terms of density, accuracy, range, and output rate. Moreover, to get a clear understanding of the amount of information included in the acquired data, a qualitative comparison between the raw points corresponding to a pedestrian standing at a distance

of about 55 m from the sensors will be performed. Finally, to complete the comparison, other aspects related to the applicability in the automotive market, such as price and size will be covered as well.

A. High-definition LIDAR

Many intelligent vehicles implementations [1], [3], [9] are based on the Velodyne HDL-64E S2 [10] high-definition LIDAR unit, depicted in Fig. 2.

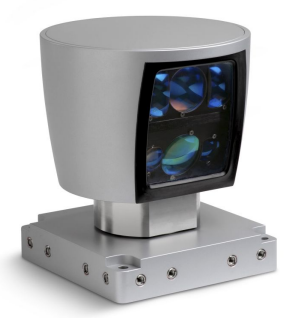


Fig. 2. A Velodyne HDL-64E S2 unit.

Some key figures regarding the device are presented in Tab. I.

TABLE I
HDL-64E S2 SPECIFICATIONS

Horizontal FOV	360°
Horizontal resolution	0.09°
Vertical FOV	26.8°
Vertical resolution	0.4°/ 64 planes
Range	50-120 m (reflectivity dependent)
Measurement error	< 2 cm
Frequency	up to 15 Hz
Shutter	rolling
Mechanical	25 cm tall cylinder, 20 cm ∅, 13 Kg, 900 RPM
Price	75,000 $

When spinning at 10 Hz the unit produces about 2.5 million distance estimations per second, effectively allowing for an accurate reconstruction of the vehicle surroundings. Sensor coverage depends on the mounting position, which directly influences the size of the blind area around the vehicle, the fixed angular resolution, and the reflectance of the measured targets.

Rolling shutter causes a deformation in the resulting data, especially if the sensor is mounted on a moving platform: when traveling straight at 10 m/s (i.e. 36 km/h) and operating at 10 Hz, the reference system moves 1 m between the start and the end of the scan, producing even larger errors when other objects move as well.

B. Stereo camera system

When it comes to the setup of a stereo vision system there are many degrees of freedom involved, since imagers, optics, and processing platforms selection offer a number of different trade-offs. As a reference, we consider the setup presented in [11], since its characteristics are similar to those commonly found in autonomous and intelligent vehicles. Moreover, this vehicle features a HDL-64E unit, and the resulting synchronized LIDAR and video data is made publicly available. In that setup, the sensors employed are two Sony ICX267 1/2" CCDs at a resolution of 1382×512 pixels, with 4.2 mm lenses, and a baseline of 0.54 m.

Many algorithms do exist that achieve dense real-time stereo reconstruction. One of the best performing is currently the so-called Semi-Global Matching, or SGM [12] approach, which has been used to produce the data in the following.

Processing platforms can also vary greatly, including traditional SIMD-capable CPUs [13], newer-generation GPUs [14], or programmable hardware, typically FPGAs [15]. Whatever the system, running dense stereo reconstruction at high resolution on commodity hardware would hardly meet the 10 Hz limit which is usually considered the lower bound for autonomous navigation. However, a commonly used technique is to keep full-resolution input images, and only perform the computationally intensive part of the algorithm on a fraction of the pixels, thus significantly reducing the processing time while retaining the original measurement accuracy [16].

Tab.II summarizes some of the key performance values of the considered stereo-based system.

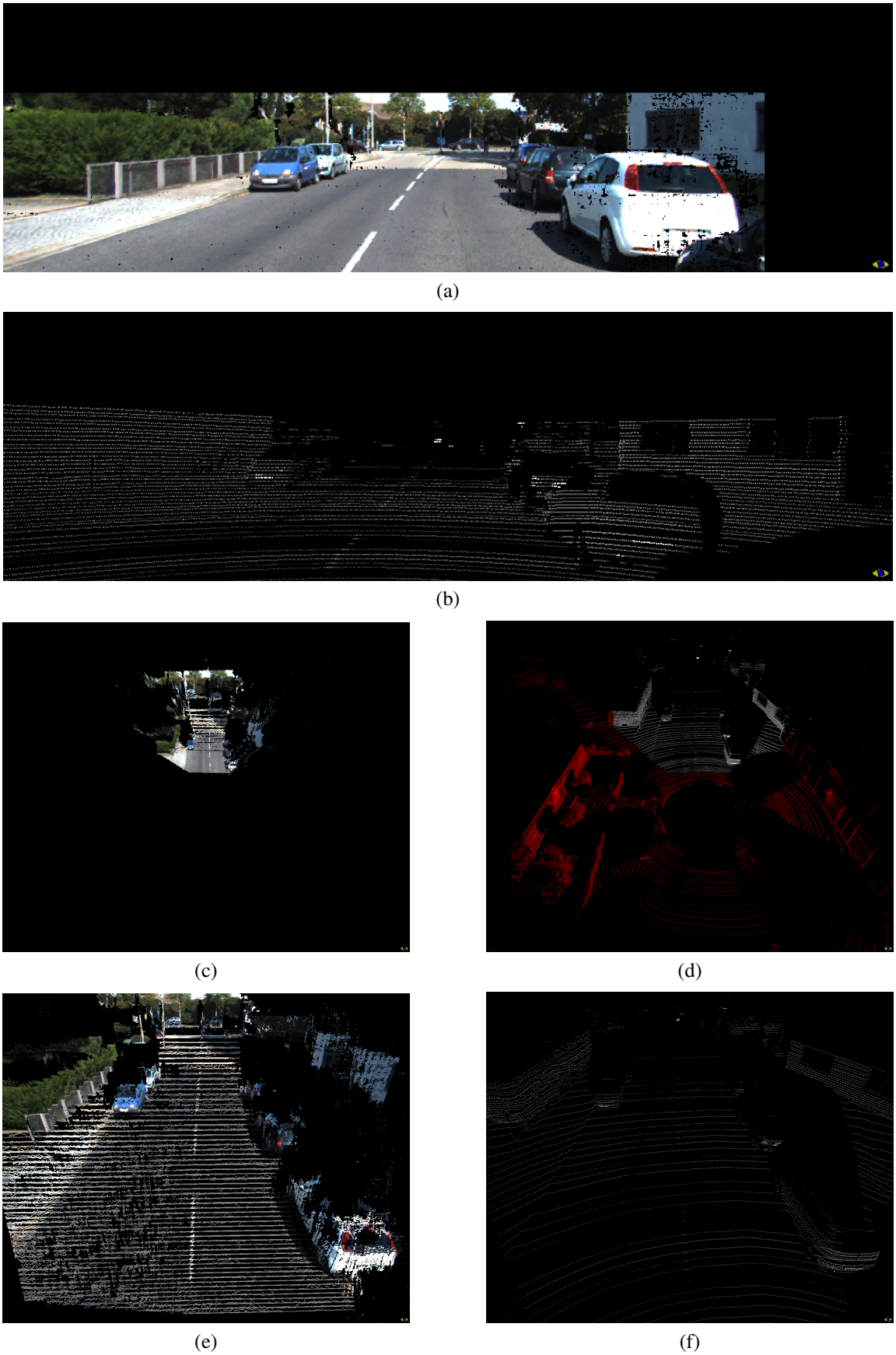
TABLE II
STEREO SAMPLE SYSTEM SPECIFICATIONS

Horizontal FOV	80°
Horizontal resolution	1232 px
Vertical FOV	29°
Vertical resolution	375 px
Max disparity value	127
Matching accuracy	0.25 px
Algorithm	SGM
Density	85 %
Range	55 m
Measurement error	0.01 m@5 m 1.9 m@55 m
Frequency	up to 10 Hz
Shutter	global
Mechanical	two 5×5 cm sensors 54 cm apart, control unit
Price	2,500 $ with COTS hardware

Running in this configuration at 10 Hz a stereo unit produces about 3.7 million distance estimations per second. Clearly, being an indirect measure, the noise affecting the measurements is higher than with the LIDAR technology; however, thorough benchmarks [11] show that in typical automotive environments the amount of bad values is around 4.5 % with a disparity error threshold of 2 pixels. This leads to around 1.4 million correct measurements per second, which is comparable to what can be obtained with the analyzed LIDAR unit.

C. Data evaluation

To obtain a better understanding of each sensor capabilities, a sample of the 3D data generated in an urban setting [11] is presented in Fig. 3. It is easy to see from Fig. 3 c,d that a single LIDAR unit mounted on top of the vehicle is enough to cover the whole area, while a single



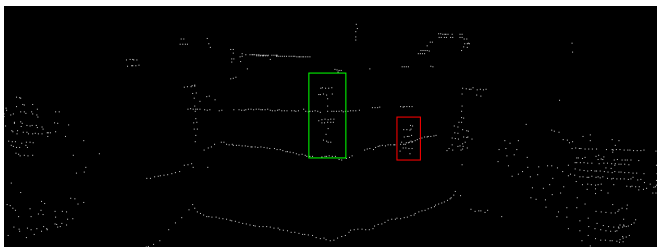
(a) (b) (c) (d) (e) (f)

Fig. 3. Qualitative comparison between LIDAR and stereo-generated 3D point clouds. a) stereo and b) LIDAR data as seen from the right camera perspective, with black areas corresponding to image pixels not being evaluated; c), d) bird's eye view of the scene, with red points in d) representing values seen from the LIDAR but not from the stereo system; e), f) a closeup of the area in front of the vehicle. Data courtesy of The KITTI Vision Benchmark Suite, available online at http://www.cvlibs.net/datasets/kitti

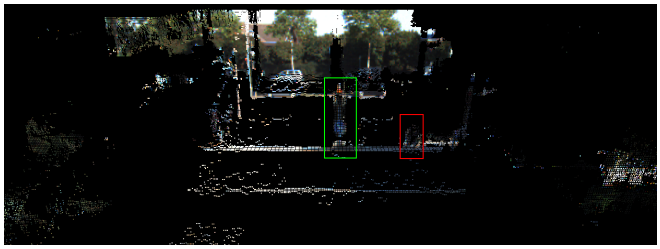
stereo system can only monitor the front portion of the scene. A closer inspection (Fig. 3 a,b,e,f) confirms that stereo vision produces much denser information, although with less and less accuracy as distance increases.

One of the most critical tasks to perform when driving is pedestrian detection. Its results must be available as soon as possible to allow for a safe maneuvering; Fig. 4 a) highlights in red a pedestrian walking across the street 50 m ahead of the vehicle. Approximating it with a rectangular target 1 m wide (because of the legs spread walking) and 1.7 m high, the LIDAR unit detects around 20 points over a theoretical maximum of 60, while the stereo systems maps almost all of the 250 points it frames. The same happens for the traffic light and road sign marked in green, which can be clearly distinguished in the stereo data, but are much harder to spot and classify in the LIDAR scan.

(a)



(b)

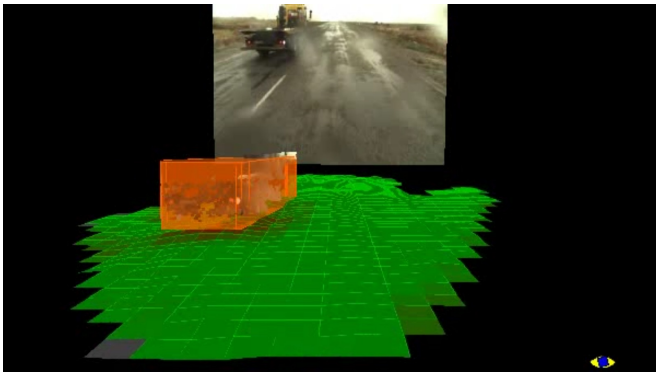


(c)

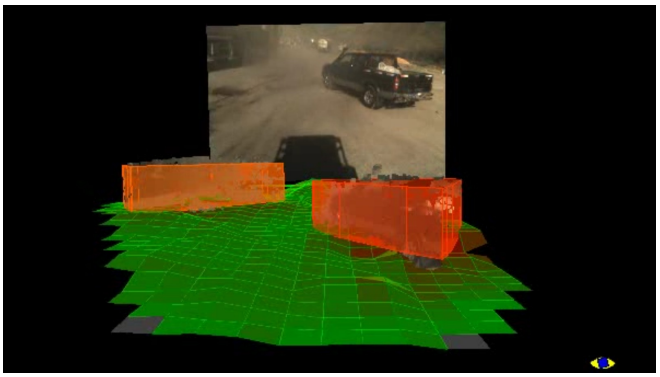
Fig. 4. Targets detection and classification. a) highlighted in green, a traffic light and in red, a pedestrian both at a distance of 50 m as seen from the right camera. The same targets are also shown in b) the LIDAR scan and c) the stereo 3D point cloud.

Some challenges however still remain when using computer vision: night-time operation requires the use of dedicated hardware, being it thermal cameras or powerful illumination systems; challenging lighting conditions should also not be overlooked, although they can be handled with the use of smart exposure control algorithms, or very high dynamic range cameras [17]. Also, field of view and measuring precision for a stereo system are lower than those of a LIDAR unit, but can be both improved by fusing together more narrow-viewing stereo systems.

Anyway, camera-based 3D mapping already has a number of features which are making it very attractive for automotive use: the cost is only a tiny fraction of the price of a typical LIDAR device, and can offer much better integration options within the vehicle. There are no moving parts, which makes it far more robust to vibrations; also there is no risk of interference when many units are operating close to each other, as it would happen during a traffic jam. Smoke, rain and dust are also less critical, since they must be present in significant amounts to hinder the detection capabilities of a stereo system, as can be seen in Fig. 5.



(a)



(b)

Fig. 5. Environment mapping in challenging scenarions: a) rain drops and b) dust do not prevent correct stereovision-based ground and obstacles detection, even if they cover a significant portion of the image.

On the algorithmic side, images provide a greater amount of information in the texture domain, which eases a lot tasks such as classification and pattern recognition, allowing at the same time simultaneous extraction of position and speed of each point in the space [18], which makes scene understanding both easier and more robust.

III. FUTURE PERSPECTIVES

The 2005 and 2007 DARPA Challenges saw high performance LIDAR sensors as a key factor for winning teams success, because of their performance and reliability; as a

matter of fact their appeal lasted, and similar design choices are still being made by top players in the field of autonomous and intelligent vehicles.

During the same period, however, a totally different approach, based on stereo-vision gained a renewed interest. This kind of technology successfully powered the TerraMax vehicle, and later on the autonomous vans employed during the 13000 km VIAC expedition [19], showing its viability as a first-class citizen in the field of environment perception.

Predicting which technology will eventually become the most widespread one is hard. Too many factors beyond pure measuring performance will influence their diffusion, however it is good to see two alternative approaches compete and evolve to provide advanced perception capabilities for next generation vehicles.

IV. ACNOWLEDGMENTS

This work has been supported by the European Research Council (ERC) within the Open intelligent systems for Future Autonomous Vehicles (OFAV) Advanced Investigators Grant (n. 228045) and its associated ERC 3DV Proof-of-concept.

REFERENCES

[1] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, vol. 25, no. 9, pp. 569–597, Sept. 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.v25:9

[2] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webster, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, vol. 25, no. 8, pp. 467–492, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.v25:8

[3] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, vol. 25, no. 9, pp. 615–639, Sept. 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.v25:9

[4] Y.-L. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, "Terramax™: Team oshkosh urban robot," *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, vol. 25, no. 10, pp. 841–860, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.v25:10

[5] L. Bombini, S. Cattani, P. Cerri, R. I. Fedriga, M. Felisa, and P. P. Porta, "Test bed for Unified Perception & Decision Architecture," in *Procs. 13th Int. Forum on Advanced Microsystems for Automotive Applications*, Berlin, Germany, May 2009.

[6] D. Braid, A. Broggi, and G. Schmiedel, "The TerraMax Autonomous Vehicle," *Journal of Field Robotics*, vol. 23, no. 9, pp. 693–708, Sept. 2006.

[7] Y.-L. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, "TerraMax: Team Oshkosh Urban Robot," in *The DARPA Urban Challenge, Autonomous Vehicles in City Traffic*, ser. Springer Tracts in Advanced Robotics, M. Buehler, K. Iagnemma, and S. Singh, Eds. Springer-Verlag Berlin Heidelberg, Nov. 2009, pp. 595–622, iSBN: 978-3-642-03990-4.

[8] http://www.valeo.com/.

[9] F. Saust, J. Wille, B. Lichte, and M. Maurer, "Autonomous vehicle guidance on braunschweig's inner ring road within the stadtpilot project," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 169 –174.

[10] http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx.

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

[12] H. Hirschmüller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information," in *Intl. Conf. on Computer Vision and Pattern Recognition*, vol. 2. San Diego, CA, USA: IEEE Computer Society, June 2005, pp. 807–814.

[13] A. Broggi, M. Buzzoni, M. Felisa, and P. Zani, "Stereo obstacle detection in challenging environments: the VIAC experience," in *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, San Francisco, California, USA, Sept. 2011, pp. 1599–1604.

[14] C. Pantilie and S. Nedevschi, "Sort-sgm: Subpixel optimized real-time semiglobal matching for intelligent vehicles," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 3, pp. 1032 –1042, march 2012.

[15] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation," in *Embedded Computer Systems (SAMOS), 2010 International Conference on*, july 2010, pp. 93 –101.

[16] S. Hermann, S. Morales, and R. Klette, "Half-resolution semi-global stereo matching," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 201 –206.

[17] http://www.new-imaging-technologies.com.

[18] C. Rabe, T. Müller, A. Wedel, and U. Franke, "Dense, robust, and accurate motion field estimation from stereo image sequences in real-time," in *Computer Vision - ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin / Heidelberg, 2010, vol. 6314, pp. 582–595.

[19] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, R. I. Fedriga, M. Felisa, L. Gatti, A. Giacomazzo, P. Grisleri, M. C. Laghi, L. Mazzei, P. Medici, M. Panciroli, P. P. Porta, P. Zani, and P. Versari, "VIAC: an Out of Ordinary Experiment," in *Procs. IEEE Intelligent Vehicles Symposium 2011*, Baden Baden, Germany, June 2011, pp. 175–180, iSSN: 1931-0587.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session II

Multiple Vehicles/Robots & Interaction

- **Title: Multiple Robots in a Cooperating Task: Exploration and Mapping**
  Authors: A.M. Neto, P. Rosa, T.E. Alves de Oliveira and P.C. Pellanda

- **Title: Dynamic Obstacle Avoidance Strategies using Limit Cycle for the Navigation of Multi-Robot System**
  Authors: A. Benzerrouk, L. Adouane and P. Martinet

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Multiple Robots in a Cooperative Task: Exploration and Mapping.

Adão de Melo Neto, Paulo Fernando Ferreira Rosa, Thiago Eustaquio Alves de Oliveira, Paulo César Pellanda

Abstract— **In this work is investigated the exploration of an environment with multiple vehicles using a strategy based on occupancy grids and a technique of simultaneous localization and mapping (SLAM). The exploration strategy uses concepts of costs and utility from frontier-cells. Besides, the used SLAM method is based on a FastSLAM algorithm with landmarks extracted from visual sensors and a features map common to vehicles. Both activities - location of the vehicles and exploration of the environment - are coordinated by a central agent. The results show that when two vehicles can communicate with a central agent building a features map common to vehicles, the exploration task becomes more efficient than that performed with dedicated maps, because the accuracy of vehicle position and orientation is increased with the use of an even number of particles. In this paper we also present and evaluate the implementation of the approach in a real environment.**

I. Introduction

Integrated exploration of an environment is a high-level activity where methods of exploration, locating, mapping and navigation have to be combined so that autonomous vehicles be able to map the environment maximizing gains (e.g., accuracy of position and orientation - pose - of the vehicles) and minimizing costs (e.g., time spent in the exploration).

SLAM is a robotic research field related to the vehicle ability to locate itself and obtain a features map of landmarks (something that can be easily detected and described) of the environment. Localization and mapping are interdependent and related functionalities since if, on the one hand, it is necessary to know the vehicle location to build a map of the environment in which it is situated, on the other, an environment map is needed to locate the vehicle. In many cases, for vehicle location, is used the combination of a GPS (global positioning system) with an IMU (inertial measurement unit). However, the information quality obtained through this system may be affected by the number of satellites in view and electromagnetic interference as well as by odometry errors. As result, they have a low precision for navigation in an indoor environment. The SLAM problem can be stated as follows [1]: given an autonomous vehicle within an unknown environment and, using only observations relative to detectable landmarks in the environment in relation to the vehicle, build a features map for those landmarks and simultaneously compute an estimation of the vehicles location based on this map. In this work is used a SLAM technique known as FastSLAM [2] to locate the vehicles. Since a SLAM algorithm is ensuring the location of a vehicle, an occupancy grid can be simultaneously generated to assist the environment exploration [9], considering that the workspace has a limited size.

The objective of this paper is to show that we can explore an indoor environment with multiply vehicles in a efficient way when we are using a FastSLAM algorithm so as to locate them, considering landmarks extracted through of SIFT algorithm [13], a computer vision algorithm used to detect and describe features in images, and a features map common to vehicles. In this approach, the vehicles increase the accuracy in the pose in relation the exploration with the use of dedicated maps. The improvement in accuracy, due to the use of a common features map, is obtained using the same number of particles (a particle represents an estimate of the pose of the vehicle and of the features map of landmarks detected - Eq.5). The main contribution of this paper is to show that the approach increases the accuracy in the pose of the vehicles, as well as present and evaluate the implementation of the approach using vehicles pioneer 3DX with sensors LMS-200 and LMS kinect [15]. Section II discusses some works related to SLAM. We treat FastSLAM considering one and two vehicles in sections III and IV. Section V focuses on exploration strategy, and the section VI describe experiments and an evaluation of the approach. We explain our conclusions in the section VII.

II. Related Works

Several studies on mapping have already been published. Some approaches estimate the vehicles pose only using odometers, which often leads to inaccuracy. Others use SLAM techniques (EKF-SLAM [1] and FastSLAM [2]) with extraction of features from raw data provided by laser and sonar sensors ([3]). In [4] a Rao-Blackwellized particle filter [2] is applied to estimate simultaneously the map and the path of a single vehicle. In the mentioned work, SIFT features are used as landmarks in the environment and extracted using a pair of stereo cameras. SLAM approaches with multiple vehicles can be grouped into two solutions. In the first group, each vehicle estimates its own individual map using its observations and, at a later stage, a common map is formed by fusing the individual maps of the vehicles. In the second one, the estimation of the trajectories and the map are made jointly. A single map is computed simultaneously by using the observations of all the vehicles. The work presented in [6] and [8] can be classified in the first group. In [6], each robot builds its own map and at the same time continuously attempts to localize in the maps built by

This research is carried out under a grant from FAPERJ (Foundation for Research of the State of Rio de Janeiro).

Defense Engineering Graduate Program Military Institute of Engineering, Rio de Janeiro, Brazil, Tel: 55-21-25467092, E-mails: adao@ime.eb.br, rpaulo@ime.eb.br, thiago.eustakio@gmail.com, pellanda@ime.eb.br.

other vehicles using particle filters. The approach can cope with the situation where the initial locations of the vehicles are unknown, however, the fusion of the individual maps is computationally expensive. [8] proposes an algorithm for multiple vehicles based on [2] where they can start from unknown pose in advance. In this approach, each vehicle builds and maintains its own map. When they are on the same line of sight, the maps are fused. The approach presented in [7] and [5] belong to the second group. [7] uses an extended Kalman filter (EKF) to estimate a state vector formed by the poses of all the vehicles and a set of 2D landmarks. The vehicles obtain observations and construct a single unified map using the update equations of the classical EKF [1]. The initial positions of the vehicles must be known in advance and the data association is assumed to be known. In this case, the main drawback stems from the fact that a single hypothesis about the pose of vehicle is maintained. [5] presents an algorithm based on [2] where the map is common for vehicles and it is assumed they have initial pose known in advance. The authors show, through experiments conducted in the *Matlab* and the observation of visual landmarks, that the approach proposed is suitable for a small groups of vehicles. As regards the exploration strategies, we highlight the approaches described in [10] that uses concepts of costs and utility from frontier-cells.

This work shows that in the context of an integrated exploration with multiply vehicles (we used two vehicles for validation), that the FastSLAM approach with a common map increases the accuracy in the pose of vehicles. The FastSLAM approach differs from EKF-SLAM for multiple hypothesis data association. Four experiments were conducted. In the first a *Player/Stage* simulator was used to test the complete approach, considering each 3D visual landmark represented by an image that has only a SIFT feature. In the second one, the data association is analyzed using images taken with a *kinect* visual sensor, an active stereo sensor which reduces the correspondence problem in a stereo pair and has a effective range of 0.8 to 3.5 meters. In the third and fourth experiments a Pioneer 3DX vehicle with laser and *kinect* sensors is used. The main contribution of this paper is to show that the above approach increases the accuracy in the pose of the vehicles presenting and evaluating the implementation of the approach using a Pioneer 3DX vehicle with sensors laser and *kinect*.

III. FastSLAM

FastSLAM [2] decomposes the SLAM problem into a vehicle localization problem, and a collection of landmark estimation problems that are conditioned to the vehicle pose estimation. Let u_t be a control action responsible for the exchange of state of a vehicle at time t. In robotics, the pose χ_t of a vehicle and the observation z_t of a landmark θ_j are modeled by probabilistic laws

$$p(\chi_t|\chi_{t-1}, u_t) \text{ and } p(z_t|\chi_t, \theta_j) \tag{1}$$

with values sampled by functions usually nonlinear in their arguments (h and g) with a Gaussian noise added with mean 0 and, respectively, covariance Q_t and R_t:

$$\chi_t = h(u_t, \chi_{t-1}) + N(0, Q_t) \text{ and } z_t = g(\chi_t, \theta_j) + N(0, R_t) \tag{2}$$

FastSLAM estimates the posterior probability distribution of the vehicle's path $\chi^t = \{\chi_1, \ldots, \chi_t\}$ and the map Θ (Eq.3) considering the observations $z^t = \{z_1, \ldots, z_t\}$, control actions $u^t = \{u_1, \ldots, u_t\}$ and associations $a^t = \{a_1, \ldots, a_t\}$ between the features of landmark that were observed and the features of landmarks in the map

$$p(\chi^t, \Theta|z^t, u^t, a^t) \tag{3}$$

It is shown that if the path χ^t is known, then the position of landmarks θ_i in Θ are conditionally independent, which allows to factor the problem of estimating the posterior probability distribution of χ^t and Θ as a product of simple terms (Eq.4).

$$p(\chi^t, \Theta|z^t, u^t, a^t) = \underbrace{p(\chi^t|z^t, u^t, a^t)}_{Path} \prod_{n=1}^{N} \underbrace{p(\theta_n|\chi^t, z^t, u^t, a^t)}_{Landmark} \tag{4}$$

Posterior probability distribution of χ^t is estimated using a particle filter [11] and the posterior probability distribution for the N landmarks θ_i of each particle are estimated by N Extended Kalman filters (EKF) [11] conditioned to the path χ^t. The particle filter represents the distribution using a set $S_t = \{S_t^{[1]}, \ldots, S_t^{[M]}\}$ of particles

$$S_t^{[m]} = \left[\chi_t^{[m]}, \underbrace{\psi_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, d_1^{[m]}, ..., \psi_{n,t}^{[m]}, \Sigma_{n,t}^{[m]}, d_n^{[m]}, ...}_{\Theta^{[m]}}, w_t^{[m]} \right] \tag{5}$$

where, $\psi_{n,t}^{[m]}$ and $\Sigma_{n,t}^{[m]}$ are the mean value and covariance for the coordinates of θ_n conditioned to the path $\chi^{t,[m]}$; $d_n^{[m]}$ is the descriptor of its features and $w_t^{[m]}$ is the weight of the particle. When the vehicle makes an observation it must update its map. Since the sensors are prone to errors, each information embedded in the map may have a certain amount of uncertainty. This inaccuracy can lead to errors in data association and in update. Update errors can be minimized through successive observations and, therefore, in order to improve accuracy in the map, it is necessary to *close the loop*, i.e., observe again a landmark previously observed. This error is also due to the linearization performed by EKF. Error in data association, caused by the erroneous association (correspondence) of features of a landmark on map and features of a landmark observed can be avoided using a robust set of features, as is the case of SIFT features, whose descriptor, a vector of dimension 128 is invariant to scale and rotating of the image and partially invariant to changes in lighting and 3D viewpoint of the camera. The FastSLAM algorithm used here is shown in sequence. We consider an observation $v_t = \{z_t, d_t\}$ where d_t is the SIFT descriptor of a visual landmark and $z_t = [d_x, d_y, d_z]^T$ the distance between the landmark and the vehicle (Fig.1).

FastSLAM (S_{t-1}, z_t, d_t, u_t)

1. $S_t = S_{aux} = \emptyset$
2. For each particle $S_{t-1}^{[m]}$ in S_{t-1}
   1. Sample $\chi_t^{[m]} \sim p(\chi_t | \chi_{t-1}^{[m]}, u_t)$
   2. For each landmark $\theta_n^{[m]}$ in map $\Theta^{[m]}$ compute
      1. $E_n^{[m]} = Mahalanobis(d_n^{[m]}, d_t) = (d_n^{[m]} - d_t)(d_n^{[m]} - d_t)^T$
      2. $p_{n,t}^{[m]} = f(z_t, \theta_n^{[m]})$
   3. End For
   4. j = Find $p_{n,t}^{[m]} \geq P_0$
   5. $a_t = argmin_j \; E_j^{[m]}$
   6. If $E_{a_t}^{[m]} \leq E_0$
      1. Update of landmark $\theta_{a_t}^{[m]}$
      2. $w_t^{[m]} = p_{a_t,t}^{[m]}$
   7. Else
      1. New landmark in $\Theta^{[m]}$
   8. End If
   9. $S_t^{[m]} \Rightarrow S_{aux}$
3. End For
4. id =Best_particle (S_t)
5. Resampling(S_t)
6. Return S_t

For each particle $S_{t-1}^{[m]}$ in S_{t-1}, the vehicle movement model is sampled (step 2.1, Eq.1, Eq.2). On the other hand, for each landmark $\theta_n^{[m]}$ in the map $\Theta^{[m]}$ is computed the distance $E_n^{[m]}$ between the SIFT descriptor d_t of the landmark observed and the descriptor $d_n^{[m]}$ of $\theta_n^{[m]}$ (step 2.2.1), and the quality of the association of $\theta_n^{[m]}$ with z_t through a function of the FastSLAM (step 2.2.2). The data

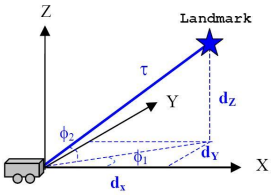


Fig. 1. Observation Model of a visual landmark 3D.

association chooses a set j of landmarks with association probability greater than P_0 (step 2.4) and select, in this set, the landmark that minimizes $E_j^{[m]}$ (step 2.5). If $E_{a_t}^{[m]}$ is less than a threshold value E_0 (step 2.6), the data association is considered correct and the estimation of the coordinates of landmark $\theta_{a_t}^{[m]}$ is updated with equations of the EKF (step 2.6.1). Otherwise, a new landmark is created on the map (step 2.7.1)[5]. The weight of the particle w_t corresponds to quality of the association $p_{t,a_t}^{[m]}$ of landmark $\theta_{a_t}^{[m]}$ which is associated with the observation z_t (step 2.6.2). The best particle - which corresponds to a estimate of location - has the greatest weight (step 4). In resampling step (step 5), particles with higher weight $w_t^{[i]}$ are replicated.

The FastSLAM algorithm in the case of one vehicle requires time MN. This is because M particles need to be processed, whereas, for each particle the data association needs to iterate over the N landmarks in the map. However, if each particle is stored in a *kd-tree* structure with dimension of the SIFT descriptor (128), the research in each structure by a list of landmarks (nearest neighbors at a distance E_0) costs $logN$, which accelerates the data association, implying a time of $MlogN$ (step 2.2). In this list, is chosen those which have association probability greater than P_0 (step 2.4).

The success obtained with SLAM has motivated the research on SLAM with multiple vehicles, as discussed in the next section.

IV. FastSLAM with Multiple Vehicles

When multiple vehicles have the possibility to communicate with a central coordinator agent, they can work together to reduce the exploration time and allows a cooperation in observing landmarks from the environment.

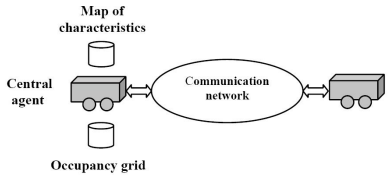


Fig. 2. Scheme of exploration with two vehicle in which a central agent is responsible for building the map (localization) and the grid (exploration).

If k vehicles explore the environment building a common map (Fig.2), and at instant t the vehicle (i), in pose $\chi_{t,(i)}$, performs a single observation $z_{t,(i)}$, the posterior probability distribution of the path $\chi_{(1:k)}$ of k vehicles and the map Θ can be estimated from the following function [5]:

$$p(\chi_{(1:k)}^t, \Theta | z_{(1:k)}^t, u_{(1:k)}^t, a^t) = \underbrace{p(\chi_{(1:k)}^t | z_{(1:k)}^t, u_{(1:k)}^t, a^t)}_{Path} \prod_{n=1}^{N} \underbrace{p(\theta_n | \chi_{(1:k)}^t, z_{(1:k)}^t, u_{(1:k)}^t, a^t)}_{Landmark}$$

where $\chi_{(1:k)}^t = \{\chi_{(1)}^t, \ldots, \chi_{(k)}^t\}$, $u_{(1:k)}^t = \{u_{(1)}^t, \ldots, u_{(k)}^t\}$ and $z_{(1:k)}^t = \{z_{(1)}^t, \ldots, z_{(k)}^t\}$ are respectively the set of paths, actions, and observation of k vehicles and $a^t = \{a_1, \ldots, a_t\}$ is the data associations history. Posterior probability distribution of $\chi_{(1:k)}^t$ is estimated using k particles filters. On the other hand, the posterior probability distribution of N landmarks θ_i, corresponding to each particle, is estimated by kN independent extended Kalman filters, conditional on the paths $\chi_{(1:k)}^t$. Since the map is common to vehicles, k particle filters produce the same set S_t of particles

$$S_t^{[m]} = \left[\chi_{(1:k),t}^{[m]}, \underbrace{\psi_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, d_1^{[m]}, \ldots, \psi_{N,t}^{[m]}, \Sigma_{N,t}^{[m]}, d_N^{[m]}}_{\Theta^{[m]}}, w_t^{[m]} \right] \quad (6)$$

Note that the state to be estimated is composed by the pose $\chi_{t,(1:k)}$ of the k vehicles. An algorithm for multiple vehicle SLAM is shown as follows [5]. The routine FastSLAM* in algorithm corresponds to the FastSLAM algorithm of the section III considering only steps 1-3.

FastSLAM with Multiple (two) Vehicles

1. $S_t = \emptyset$
2. For $t = 1$ to End do
   1. $[z_{t,(1)}, d_{t,(1)}, z_{t,(2)}, d_{t,(2)}]$ = Observations()

2.$[S_t, w_{t,(1)}] = FastSLAM^*(S_{t-1}, z_{t,(1)}, d_{t,(1)}u_{t,(1)})$
3.$[S_t, w_{t,(2)}] = FastSLAM^*(S_{t-1}, z_{t,(2)}, d_{t,(2)}u_{t,(2)})$
4.$w_t = w_{t,(1)}w_{t,(2)}$
5.$S_t =$ Resampling (S_t, w_t)
3. End For

This algorithm consider that the vehicles begin exploration from a pose known by the central agent. Considering that vehicles start from nearby positions, the pose relative can be obtained by laser sensor. Since the map is common to vehicles, for each particle $S_t^{[m]}$ defined in equation 6, k weights are calculated and the total weight associated with particle $S_t^{[m]}$ is defined as

$$w_t = \prod_{i=1}^{k} w_{t,(i)}^{[m]} \tag{7}$$

FastSLAM algorithm in the case of k vehicles require time $kMlogN$.

We present below the adopted exploration strategy.

V. Exploration Strategy

In an exploration activity, the path to be followed by a vehicle must be controlled for the sake of efficiency. In the considered approach [10], a target - frontier-cell of an occupancy grid [9] - is chosen according to a function that evaluates the cost of navigation and the utility of the target.

A. Costs

In order to determine the cost of reaching the current frontier-cells, we compute the optimal path from the current position of the vehicle to all the frontier-cells based on a deterministic variant of the *value iteration*, a popular dynamic programming algorithm [14]. It is considered that the cost for traversing a grid cell (x, y) is proportional to its occupancy value $P(occ_{xy})$. The minimum-cost path is computed using the two steps below:

1) **Initialization.** The grid cell that contains the location of vehicle is initialized with 0, and all others with ∞

$$V_{x,y} \leftarrow \begin{cases} 0, & \text{if (x,y) is the vehicle position} \\ \infty, & \text{otherwise} \end{cases}$$

2) **Update loop.** For all grid cells (x, y) do

$$V_{x,y} \leftarrow min\{V_{x+\Delta x,y+\Delta y} + \sqrt{\Delta x^2 + \Delta y^2}P(occ_{x+\Delta x,y+\Delta y})\} \mid \Delta x, \Delta y \in \{-1, 0, 1\}, P(occ_{x+\Delta x,y+\Delta y}) \in [0, occ_{max}]$$

where occ_{max} is the maximum occupancy probability value of a grid cell the vehicle is allowed to transverse. This technique updates the value of all grid cells by the value of their best neighbors, plus the cost of moving to this neighbor. Here, cost is equivalent to the probability $P(occ_{x,y})$ that a grid cell (x, y) is occupied times the distance to the cell. The update rule is repeated until convergence. All in all each value $V_{x,y}$ corresponds to the cumulative cost of moving from the current position of the vehicle to (x, y).

B. Utilities of frontier-cells

If there is already a vehicle that moves to a frontier-cell, the utility of this cell must be lower for other vehicles. Let us suppose that in the beginning each frontier-cell t has the utility U_t which is equal for all frontier-cells. Then, we compute the utility $U(t_n|t_1, \ldots, t_{n-1})$ of a frontier-cell t_n given that the cells $t_1, \ldots, t_{n-1}$ have already been assigned to the vehicles $1, \ldots, n-1$ as

$$U(t_n|t_1, \ldots, t_{n-1}) = U_{t_n} - \sum_{i=1}^{n-1} P(\underbrace{||(t_n - t_i||}_{d}) \tag{8}$$

where $P(d)$ is the probability that the sensor of vehicle (a laser sensor in our case). will cover cells in distance d. According to Equation 8, the more vehicles move to a location from where t_n is likely to be visible, the lower is the utility of t_n. We compute $P(d)$ as

$$P(d) \leftarrow \begin{cases} 1 - \frac{d}{max\_range}, & if \quad d < max\_range \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where max_range is the maximum range reading provided by sensor.

C. Target Point Selection

In the selection of destinations, is considered for each vehicle i, a balance between the cost V_t^i of moving to a destination t and the utility U_t of this target [10].

The next section describes the experimental set up and some results.

VI. EXPERIMENTS AND RESULTS

Four experiments were conducted. In the first, a *Player/Stage* simulator (Fig. 3) was used to test the complete approach. In the second one, only the data association is verified through images obtained with a *kinect* sensor, an active stereo sensor, in a pre-defined path (Fig. 7) of a real environment. In the third and fourth experiments a pioneer 3DX vehicle with laser LMS-200 and *kinect* sensors is used. The results of the first and fourth experiments were obtained considering the use of dedicated maps (case "a") and common map (case "b").

A. Experiment 1

In the first experiment, a laser sensor Sick LMS-200 embedded in each Pioneer 3DX were utilized to build the occupancy grid. On the other hand, to simulate a stereo visual sensor with a field of view of 180^o, 3D landmarks represented by images were used. Each landmark contains only one SIFT feature and its coordinates $[\theta_x, \theta_y, \theta_z]^T$ were artificially assigned (Fig.1). The images are differents and therefore, the data association is known. When a visual landmark is within the field of vision, the SIFT algorithm extracts the descriptor of image and compute the distance τ and the orientations ϕ_1 and ϕ_2 of landmark in relation to the vehicle (Fig.1).

Selection of destinations was done as described in section V, where an occupancy grid with cells of size 0.5m x 0.5m

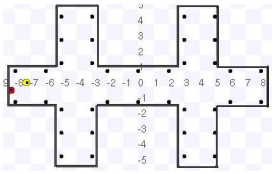


Fig. 3. Explored environment (17m x 10.5m) with vehicles in their initial pose

TABLE I

SAMPLING OF THE MOTION (FIG.4) AND OBSERVATION (FIG.1) MODEL

$N(0,\sigma)$ is a normal distribution with zero mean and standard deviation σ. Values in meters and radians.

Variable	Average error
$x_{t+1} = x_t + \Delta x_t + \overline{d}N(0,\sigma)$	0 ($\Delta x_t = 0$) or $\overline{d} = \frac{5}{100}d$ ($\Delta x_t \neq 0$)
$y_{t+1} = y_t + \Delta y_t + \overline{d}N(0,\sigma)$	0 ($\Delta y_t = 0$) or $\overline{d} = \frac{5}{100}d$ ($\Delta y_t \neq 0$)
$\varphi_{t+1} = \varphi_t + \Delta\varphi_t + \overline{g}N(0,\sigma)$	$\overline{g} = \frac{3\pi}{180}$
$\tau = \tau + \overline{\tau}N(0,\sigma)$	$\overline{\tau} = 0.1$
$\phi_i = \phi_i + \overline{\phi_i}N(0,\sigma)$	$\overline{\phi_i}$

was used. After selecting the destination, the path to be traveled by the vehicle was computed using *A-star* algorithm [12]. The position of the vehicle (x and y) and its orientation (φ) at each step in the occupancy grid (Fig. 4), distance τ and orientations ϕ_1 and ϕ_2 (Fig.1) were sampled according to the table I.

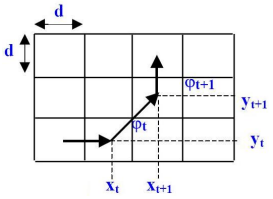


Fig. 4. Motion model used

In the experiment, the following parameters were adopted: $\sigma = 0.5$, $\overline{\phi_i} = \frac{0.1\pi}{180}$ rad, $d = 0.5$ m, $E_0 = 0$, $P_0 = 0.9$, 2000 particles and range of the visual sensor of 3.5 meters. Since each landmark has only one (unique) SIFT descriptor, we did $E_0 = 0$. Our goal is to show that in the context of a perfect data association, the approach with common map increases the accuracy in the pose of the vehicles. Figure 5 and 6 show the trajectories traveled by the vehicles in tasks of exploration.

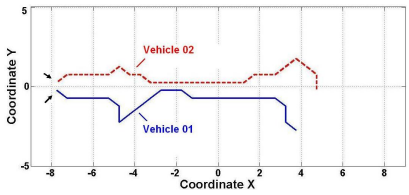


Fig. 5. Trajectory traveled by vehicles in an exploration task: vehicle 01 (continuous line) and vehicle 02 (dashed line). The arrows indicate the initial position of vehicles. Total number of steps per vehicle: 28.

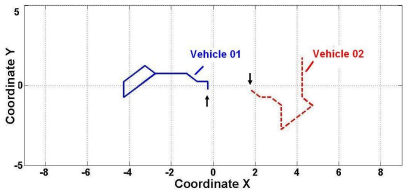


Fig. 6. Trajectory traveled by vehicles in an exploration task: vehicle 01 (continuous line) and vehicle 02 (dashed line). The arrows indicate the initial position of vehicles. Total number of steps per vehicle: 16.

The results of the table II is related to figure 5. The results of the tables III is related to figure 6. In table II we can see that the accuracy and amount of estimates of location obtained in exploration of the case "b" (common map) increased in relation to the case "a" (dedicated maps). Note that we are using a very restrictive condition P_0 to ensure a good accuracy.

TABLE II

RESULTS IN PLAYER/STAGE SIMULATOR (FIG.5)

Error (RMS) in coordinates x,y and orientation α; amount of estimate of location obtained divided by the number of steps used in exploration; and amount of updates of landmark on the map.

Case	x	y	α	$\frac{Estimate}{Step}$	Update
a 1	0.042	0.040	**0.108**	**16/28**	**106**
a 2	0.042	0.037	**0.105**	**17/28**	**71**
b 1	0.045	0.043	**0.066**	**23/28**	**180**
b 2	0.038	0.041	**0.073**	**25/28**	**215**

Maximum and minimum values (average $\pm$ standard deviation)

Case	x_{min}	x_{max}	y_{min}	y_{max}	α_{min}	α_{max}	$\frac{Estimate}{Step}$
a 1	0.020	0.064	0.022	0.058	**0.066**	**0.150**	**57%**
a 2	0.019	0.064	0.022	0.052	**0.065**	**0.145**	**61%**
b 1	0.022	0.068	0.031	0.055	**0.037**	**0.096**	**82%**
b 2	0.025	0.050	0.029	0.053	**0.052**	**0.095**	**89%**

TABLE III

RESULTS IN PLAYER/STAGE SIMULATOR (FIG.6)

Error (RMS) in coordinates x,y and orientation α; amount of estimate of location obtained divided by the number of steps used in exploration; and amount of updates of landmark on the map

Case	x	y	α	$\frac{Estimates}{Steps}$	Update
a 1	0.027	0.032	0.062	13/16	92
a 2	0.032	0.029	0.066	13/16	82
b 1	0.026	0.032	0.069	14/16	140
b 2	0.029	0.028	0.059	15/16	143

Maximum and minimum values (average $\pm$ standard deviation)

Case	x_{min}	x_{max}	y_{min}	y_{max}	α_{min}	α_{max}	$\frac{Estimate}{Step}$
a 1	0.018	0.037	0.021	0.042	0.032	0.093	81%
a 2	0.015	0.048	0.019	0.039	0.034	0.097	88%
b 1	0.018	0.034	0.022	0.042	0.037	0.101	87%
b 2	0.017	0.043	0.022	0.034	0.035	0.084	93%

When a vehicle is exploring an environment, the accuracy of the estimated pose tends to decrease, since it depends on the estimated accuracy of the landmarks mapped. In turn, the estimated coordinates of the landmarks mapped depend

on the accuracy of the estimated pose of the vehicle, which tends to decrease as the vehicle explores the environment if it does not occur corrections through observations of landmarks mapped previously. Therefore, a greater number of updates of landmarks tends to improve the accuracy of the pose estimated by SLAM and it is better that they occur as often as possible. The SLAM allows to estimate the pose of the vehicle based on the estimation error of the odometer of the vehicle and of the model of observation. This estimate of the pose is used to correct the error in odometry of the vehicle, considering that the environment is small, since the estimate depends on the accuracy of the landmarks detected. The use of FastSLAM with common map is justified by the fact that the update of landmarks tends to happen more often, and in a shorter distance traveled by the vehicle. Even with a increase in the size of the problem to be estimated by FastSLAM (pose of two vehicles), the experiments confirms that this approach produces a good estimate of vehicle location, because a vehicle can update a landmark detected by itself or by another vehicle, so that, probabilistically, updates occur more often. Using a common map is more advantageous in the situation where a vehicle has to pass close to a region where another vehicle has already covered (Fig.5). In this approach, although the choose of best particle is made individually by each vehicle, the weight of each particle is the product of the weight of the estimates of each vehicle (Eq.7) and the resampling of particles is done based on this weight, which means that a particle updated only by a vehicle will be resampled to a less proportion. However, the SIFT algorithm allows to obtain many landmarks and the addition of more vehicles will increase the number of updates of landmarks, since this update can be done in a landmark mapped by any of the vehicles. The results of table III confirm that the FastSLAM with common map is a good estimator because when a vehicle does not update landmarks on map inserted by another vehicle (the sensor range is 3.5 meters and the path of the vehicles are different), the results (accuracy and number of estimates of location by step) are similar to FastSLAM with dedicated maps.

B. Experiment 2

In the third experiment, a path (Fig.7) in an environment containing chairs, desks, computers, so on, was travelled with a visual sensor *kinect* and, at each step, an image and a file containing the distances $[d_x, d_y, d_z]^T$ of pixels to the sensor was obtained and stored to then be used to simulate a vehicle running a FastSLAM algorithm.

The figure 8 shows the variation in number of updates on the map, considering, respectively, the variation in limits P_0 and E_0 (Fig. 8). In the simulation, the distance τ and the orientations ϕ_1 and ϕ_2 (Fig.1) for SIFT landmarks detected were obtained from the distance $[d_x, d_y, d_z]^T$. The following parameters were adopted: 1000 particles, $\sigma = 0.5$, $\overline{\phi_i} = \frac{0.1\pi}{180}$ rad, $d = 1.0$ m and effective range of *kinect* (0.8 to 3.5 meters). We can conclude that, respectively, a P_0 too high and a E_0 too low may inhibit updates. On the other hand, a value of P_0 much low with a value of E_0 much high can degrade the map and the FastSLAM algorithm. The ideal is a high value of P_0 and a value of E_0 sufficiently low.

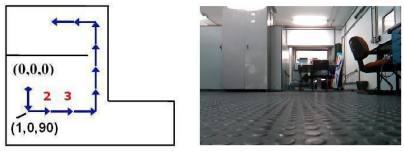


Fig. 7. Path (10 meters) in an unstructured environment containing desks, chairs, computers, and so on, traveled - of simulated manner - by the vehicle (left) and picture of the environment mentioned (right)

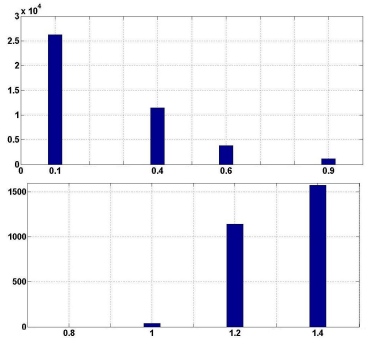


Fig. 8. Average number of updating on map in function of P_0 ($E_0 = 1.2$) - above - and E_0 ($P_0 = 0.9$) - below.

To define the value of E_0, we did a separate analysis of SIFT descriptors regardless of the effective range of the *kinect*. We obtained the descriptors of all images (3125)and compute, for each descriptor, the Mahanalobis distance to other descriptors. We found that the descriptors obtained in a same image are different, ie, all distances are different from 0. The average Mahanalobis distance was equal to 1.2. The table IV shows the percentage of distances between the descriptors of all images that are below certain limits.

TABLE IV
ANALYSIS OF DESCRIPTORS 2

Distance-Limit (E_0)	Quantity of distances	Percentage
0.1	0	0
0.3	252	0.003
0.5	6362	0.065
0.7	53588	0.550
0.9	298898	3.061
1.1	1559184	15.971
1.2	3319498	34.002

As the SIFT descriptor is partially invariant to the point of sight 3D and lighting, we did an analysis of the correspondence (association) of the descriptors of two images (Fig.9). From the results obtained and consolidated in the table V, we found that the values $E_0 = 0.4$ and $E_0 = 0.5$ provides a good combination in relation to the number of correct correspondences (large) and to number of correspondences possible (little). In that table the calculations were made considering that there are SIFT features corresponding in the two images (Fig.9). However, in the FastSLAM the correspondence (association) is made between an observed

landmark of an image and the landmarks of the feature map of a particle. Let's assume that the landmark observed does not have a correspondence on the map (is the first time what was observed), then, an association (erroneous) will occur only if the position of the associated landmark is very close to the predicted position by the particle of the FastSLAM, since we are using $P_0 = 0.9$ (a high value). Therefore, the lower the value of E_0, the better the quality of the data association, since we have a smaller number of possible correspondences (Tab.V).

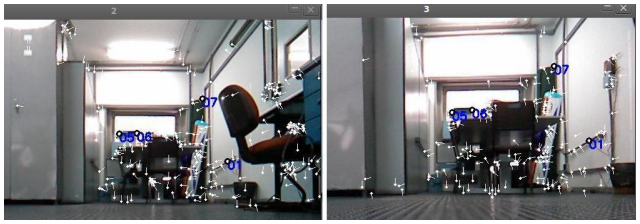


Fig. 9. Correspondence of descriptors of the images obtained in steps 2 and 3 (Fig.7) considering $E_0 = 0.4$

TABLE V
ANALYSIS OF DESCRIPTORS 3

E_0	$\frac{correct\_associations}{Total\_associations}$
0.4	$04/04 = 100\%$
0.5	$13/14 = 93\%$
0.6	$22/25 = 88\%$
0.7	$31/42 = 74\%$

C. Experiment 3

In the fourth experiment we traveled an environment (Fig.10) with a Pionner 3DX vehicle following the path defined in the figure 10. In the experiment, the following parameters were adopted: $\sigma = 0.5$, $\overline{\phi_i} = \frac{0.1\pi}{180}$ rad, 1000 particles and effective range of *kinect*.

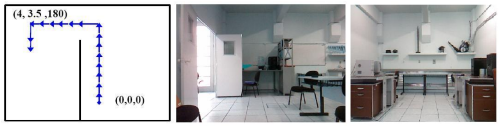


Fig. 10. Path (16 steps) travelled by Pioneer 3DX (Each step has 0.5 or 1.0 meters) and corresponding images in the poses (0,0,0) and (4,3.5,180).

In the figure 11 we have the processing time per step in function on number of landmarks on the map. The total time spent was 494 sec. Assuming that each step we expect 10 seconds for the displacement of the vehicle, the total processing time of the FastSLAM was 334 seconds (5.56 min or 21 sec/steps). If we had not used the *kd-tree* structure, the time would grow linearly with the number of landmarks on the map. In 50% of steps was obtained an estimate of location. In the figure 12 we have the higher resolution grid - *laser map* - considering in each step the best particle.

In order to assess the impact of the choice of E_0 in processing time of the FastSLAM, from the images and from

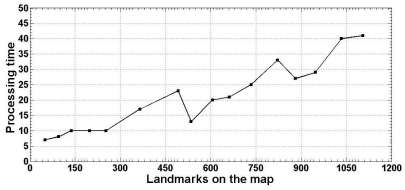


Fig. 11. Processing time per step in function on number of landmarks on the map ($P_0 = 0.9$ and $E_0 = 0.5$)

Fig. 12. Higher resolution grid - *laser map* - generated by best particle ($P_0 = 0.9$ and $E_0 = 0.5$).

the file containing the distances $[d_x, d_y, d_z]^T$ of pixels in relation to sensor that was stored during the experiment, we get the amount of estimates of location, amount of updates of landmarks and processing time (Tab.VI) considering 1000 particles. From the results we can conclude that the processing time not varied considerably.

TABLE VI
RESULTS IN FUNCTION OF E_0 CONSIDERING 1000 PARTICLES.

E_0	$\frac{Estimates}{Steps}$	Updates	Time (sec)	Time (min)
0.3	3/16	03	299	4.98
0.4	6/16	15	319	5.31
0.5	8/16	37	334	5.56
0.6	9/16	88	369	6.15

D. Experiment 4

In the fifth experiment, from the images and files containing the distances $[d_x, d_y, d_z]^T$ of the pixels in relation to *kinect* of the experiment 4, we simulate two vehicles traveling the paths described in figure 13 and using FastSLAM with dedicated maps (case "a") and common map (case "b").The following parameters were adopted: 1000 particles, $\sigma = 0.5$, $\overline{\phi_i} = \frac{0.1\pi}{180}$ rad and effective range of the *kinect*. This experiment considered in case "b" that a

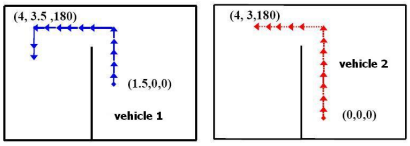


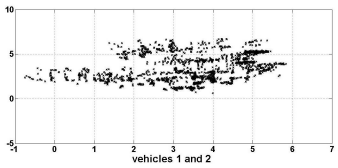
Fig. 13. Path (13 meters) traveled - of simulated manner - by vehicle 01 (continuous line) and 02 (dashed line). Each step has 0.5 or 1.0 meters.

vehicle will only insert a new landmark on the map if the region was not mapped by another vehicle (the vehicle 2 will insert landmarks only in the step 1 and 2). In this case the vehicle 2 (case "b") obtained a greater amount of estimate of location and updates of landmarks (Tab.VII). This

implies that there was observations of landmarks previously observed by vehicle 1. The figure 14 shows the landmarks map generated by the best particle. In the case "b", each particle which updated a landmark, did it in just a few of them ($P_0 > 0.9$), $N_{cm} \simeq 0.57 * (N_{dm_1} + N_{dm_2})$ and the searching of nearest neighbors selected 1.98 times more landmarks.

TABLE VII
RESULTS CONSIDERING IN FUNCTION OF $E_0 = 0.5$ AND CONSIDERING 1000 PARTICLES.

Case	vehicle	$\frac{Estimates}{Steps}$	Updates	Time (min)
a	1	05/13	38	4.10
a	2	08/13	47	3.71
b	1	05/13	54	9.98
b	2	12/13	831	9.98



Case "b" - Common map

Fig. 14. 2D Map generated by the best particle ($P_0 = 0.9$, $E_0 = 0.5$)

E. Evaluation of the approach

The results show that using the same number of particles, the exploration conducted with common map increases the accuracy in pose of vehicles. It was expected that the increase in number of vehicles used in the exploration entailed the need to increase the number of particles in order to achieve a similar pose error estimation, which turned out to be unnecessary. In both approaches, the accuracy achieved by each vehicle depends on the updates of landmarks. When using a common map to vehicles, this number tends to increase because these updates can be made also in a landmark previously mapped by another vehicle. When using a common map, all processing will get concentrated in the central agent, which should have a capacity equivalent to the number of vehicles used (two in our case) so that time spent in the exploration is similar. The approach with common map requires a time $2MlogN_{cm}$ instead of $MlogN_{dm}$ ($N_{cm} \simeq 0.5*(N_{dm_1}+N_{dm_2})$). Although there is an increased computational cost, the order of complexity remains $O(MlogN_{cm})$ and landmarks are stored in a *kd-tree* structure. Therefore, this increase can be compensated by a greater computational power in central agent or paralleling the approach with common map. The processing of landmarks observed by vehicles can be parallelized, since a particle represents an estimate of the pose of the vehicle with their respective set of estimated landmarks. It is expected that the processing time with common map and dedicated maps are approximately identical. In relation to complexity in extraction of landmarks with algorithm SIFT, we consider that it is done individually by each vehicle. Regarding the need for communication for the transmission of descriptors for the central agent, this communication is also required for transmission of upgrades of the occupation grid.

VII. CONCLUSION

In this work we show that we can explore an indoor environment with multiply vehicles in a efficient way when we are using a FastSLAM algorithm so as to locate them, considering landmarks extracted through of SIFT algorithm and features map common to vehicles. This is because the vehicles increases the accuracy in the pose, everything in relation the use of dedicated maps. The improvement is obtained with the same number of particles, which could potentially be larger, since the estimated problem is greater.

REFERENCES

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics And Automation*, Vol. 17, Issue 3, pp. 229-241.

[2] M. Montemerlo and S. Thrun (2003). Simultaneous localization and mapping with unknown data association. *Proceedings of the IEEE, International Conference on Robotics and Automation*, Vol. 2, pp. 1985-1991.

[3] C. Fulgenzi, G. Ippoliti, S. Longhi (2009). Experimental validation of FastSLAM algorithm integrated with a linear features based map. *Elsevier: Mechatronics*, Vol. 19, Issue 5, pp. 609-616

[4] A. Gil, O. Reinoso, O. Martnez-Mozos, C. Stachniss, W. Burgard (2006). Improving Data Association in Vision-based SLAM. *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* pp. 2076-2081

[5] A. Gil, O. Reinoso, M. Ballesta, M. Julia (2009). Multi-robot visual SLAM using Rao-blackwellized particle filter. *ACM: Robotics and Autonomous Systems*, Vol. 58, Issue 1, pp. 68-80.

[6] B. Stewart, J. Ko, D. Fox, K. Konolige (2003). A hierarchical Bayesian approach to mobile robot map structure estimation. *Proceedings of the Conference on Uncertainty in AI, UAI.*,

[7] J.W. Fenwick, P.M. Newman, J.J. Leonard (2002). Cooperative concurrent mapping and localization. *IEEE Int. Conf. on Robotics and Automation, ICRA*, pp. 1810-1817

[8] A. Howard (2006). Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, Vol. 25, Issue 12, pp. 1243-1256.

[9] A. A. Makarenko, S. B. Willians, F. Bourgault, H. F. Durrant-Whyte (2002). An Experiment of Integrated Exploration. *Proceedings of the 2002 IEEE-RSJ. Intl Conference on Intelligent Robots and Systems*, Vol. 1, pp. 534-539.

[10] W. Burgard, M. Moors, C. Stachniss and F. Schneider (2005). Coordinated Multi-Robot Exploration, *IEEE Transactions on Robotics, Vol. 21, Issue 3, pp. 376-386*.

[11] S. Thrun, W. Burgard and D. Fox (2005). Probabilistic robotics. *Massachusetts Institute of Technology Press*, Cambridge, MA, USA, 2005.

[12] Hart, P. E., Nilsson, N. J., Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, Vol. 4, Issue 2, pp. 100107

[13] David Lowe (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, Vol. 2, pp. 91-110

[14] R. Howard (1960). Dynamic Programming and Markov Process. *MIT Press and Wiley*.

[15] Stowers, J. and Hayes, M. and Bainbridge-Smith, A. (2011). Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. *IEEE International Conference on Mechatronics (ICM)*. pp. 358 -362

Dynamic Obstacle Avoidance Strategies using Limit Cycle for the Navigation of Multi-Robot System

A. Benzerrouk[1], L. Adouane[2] and P. Martinet[3]

[1] Institut Français de Mécanique Avancée, 63177 Aubière, France
[2] Clermont Université, Université Blaise Pascal, BP 10448, 63000 Clermont-Ferrand, France
[3] IRCCYN, Ecole Centrale de Nantes, 1 rue de la Noé,
BP 92101, 44321 Nantes Cedex 03, France
Ahmed.BENZERROUK@lasmea.univ-bpclermont.fr

***Abstract*— This paper deals with the navigation of a multi-robot system (MRS). The latter must reach and maintain a specific formation in dynamic environment. In such areas, the collision avoidance between the robots themselves and with other obstacles (static and dynamic) is a challenging issue. To deal with it, a reactive and a distributed control architecture is proposed. The navigation in formation of the MRS is insured while tracking a global virtual structure. In addition, according to the robots' perception context (e.g., static or dynamic obstacle), the most suitable obstacle avoidance strategy is activated. These approaches use mainly the *limit-cycle* principle and a *penalty function* to obtain linear and angular robots' velocities. The proposed control law guarantees the stability (using Lyapunov function) and the safety of the MRS. The robustness and the efficiency of the proposed control architecture is demonstrated through a multitude of experiments which shows the MRS in different configuration of avoidance.**

I. INTRODUCTION

Navigation of multiple mobile robots is a recurrent research subject due to a large amount of the met issues. Obstacle avoidance is among the most important ones. In fact, it is a basic action that each mobile robot has to accomplish in its environment in order to prevent collision (with walls, trees, walkers, other robots, etc.), and to insure a safe navigation.

Collision avoidance is then widely investigated in the literature for multi-robot systems. It is tackled through two main approaches. The first one considers the robots control, entirely based on path planning methods which involve the prior knowledge of the robots environment. The objective is to find the best path to all the robots in order to avoid each other while minimizing a cost function [1], [2], [3]. This method requires a significant computational complexity, especially when the environment is highly dynamic. In fact, the robot has to frequently replan its path to take environment changes into account.

Rather than a prior knowledge of the environment, reactive methods are based on local robots sensors information. At each sample time, robot's control is computed according to its perceived environment. Potential field methods [4] are the most common ones: each robot is subject to a sum of an attractive virtual force generated by the goal to reach and repulsive forces generated by the other robots and obstacles [5], [6], [7]. An other reactive method is the Deformable Virtual Zone (DVZ) [8]: every robot is surrounded by a virtual risk area. If an obstacle enters inside this DVZ, it deforms it. The aim of the generated control is then to minimize this deformation leading to avoid collision among robots [9]. The reactive methods given above suffer from local minima problems when for instance, the sum of potential forces is null, or when the deformation of the DVZ is symmetric (as the U shape obstacle). In [10], authors propose the Distributed Reactive Collision Avoidance algorithm (DRCA). This method is based on an equilibrium point which continuously pushes the robots away from each other by increasing their relative velocities. Hence, this algorithm is not suitable for the navigation in formation where robots regularly have to move with the same velocity. Generally, reactive methods do not require high computational complexities, since robots actions must be given in real-time according to the perception.

This paper deals with this last kind of methods. The studied task is the navigation in formation. It is accomplished through a distributed control architecture. This architecture was developed in [11] and permits for a group of mobiles robots to reach and maintain a specific formation. In this last work, obstacle avoidance was not addressed neither for dynamical obstacles nor to avoid other robots participating in the formation. In [11] the used strategy deals with the virtual structure. The formation is considered as a virtual rigid body and the control law for each robot is derived by defining the dynamics of this body [6], [12], [13]. Virtual structure is often associated to potential field applications since they are simple and allow collision avoidance. However, potential forces are limited, especially when the formation shape needs to be frequently reconfigured. In fact, it means that the robot is submitted to a frequently-changing number/amplitude of forces leading to more local minima, oscillations, etc. Hence, it was proposed that the robots track a virtual body without using potential forces. Since collision avoidance must stay possible despite the absence of potential fields, behavior-based concept [14], [15] was introduced. This allows to divide the task into two different behaviors (controllers): *attraction to a dynamic target*, and *obstacle avoidance*. The latter was based on limit-cycle differential equations [16].

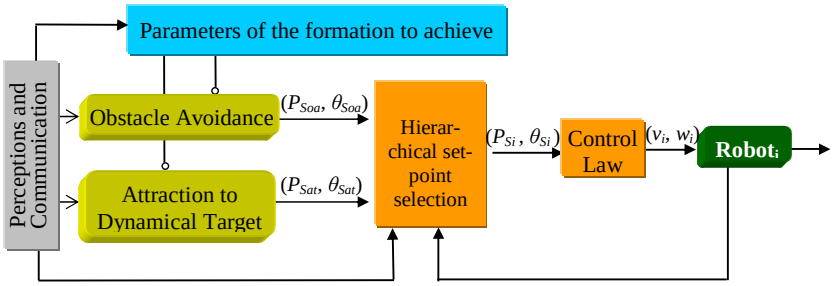


Fig. 1. The proposed architecture of control embedded in each robot.

Limit-cycle navigation was already used for obstacle avoidance [17], [18]. Limit-cycle approach allows to choose the obstacle avoidance direction (clockwise or counterclockwise) in order to rapidly join the assigned target. Here, it is proposed to extend this method to dynamic obstacles and to robots of the same system without loosing the control reactivity. Unlike most of algorithms addressing dynamic obstacles, no communication is required among the robots to accomplish the task. Avoidance is based only on the local perception of each robot. As in [19], [18] or [20] the idea is to find the best direction of avoidance. It will be seen that the velocity vector of the obstacle is sufficient to deduce this direction.

The remainder of the paper is organized as follows. Section II gives the principle of the navigation in formation and the general control architecture. Basic controllers and the the control law are given in this section. We mainly focus on the obstacle avoidance controller applied to dynamic obstacles. In section III, a *penalty function* is introduced in the linear velocity of each robot to permits to take into account the multi-robot interactions. Section IV validates the proposed algorithm with experimental results. Finally, we conclude and give some perspectives in section V.

II. Control Architecture

The used control architecture includes two controllers: Attraction to a Dynamic Target and Obstacle Avoidance. The virtual structure is built through the *Parameters of the Formation to Achieve* block (cf. Figure 1).

According to environment information collected by the *Perceptions and Communication* block (sensors) and the robot's current state, one controller is chosen thanks to the *Hierarchical Set-Point Selection* block.

The corresponding set-points (P_{S_i}, θ_{S_i}) (position and orientation) are then sent to the *Control Law* block which calculates the linear and angular velocities noted v_i and w_i respectively (cf. Figure 1).

A. Parameters of the Formation to Achieve block

This subsection briefly describes the adopted virtual structure principle. Consider N robots with the objective of reaching and maintaining them in a given formation. The proposed virtual structure that must be followed by the group of robots is defined as follow:

- Define one point which is called the main dynamic target (cf. Figure 2),
- Define the virtual structure to follow by defining N_T nodes (virtual targets) to obtain the desired geometry. Each node i is called a secondary target and is defined according to a specific distance D_i and angle Φ_i with respect to the main target. Secondary targets defined by this way have then the same orientation θ_T. However, each target i will have its linear velocity v_{T_i}. The number of these targets N_T must be $N_T \geq N$.

A cooperative strategy between the robots allows to each one to choose the closest target by negotiating it with the others thanks to Relative Cost Coefficients. To focus mainly on obstacle avoidance, this strategy is deactivated. Each robot i has then to track a predefined target i. An exemple to get a triangular formation is given in figure 2.

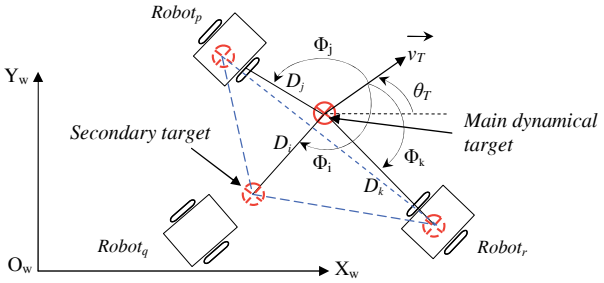


Fig. 2. Keeping a triangular formation by defining a virtual geometrical structure.

B. Attraction to a Dynamic Target controller

To remind the attraction to a Dynamic Target Controller which allows to keep the formation, consider a robot i with (x_i, y_i, θ_i) pose. This robot has to track its secondary dynamic target. To simplify notations in the following, the same subscript of the robot is given to its target. The latter is then noted $T_i(x_{T_i}, y_{T_i}, \theta_T)$ (cf. Figure 3) and the variation of its position can be described by

$$\begin{cases} \dot{x}_{T_i} = v_{T_i}.cos(\theta_T) \\ \dot{y}_{T_i} = v_{T_i}.sin(\theta_T) \end{cases} \quad (1)$$

Let's also introduce the used robot model (cf. Figure 3). Experimental results are made on Khepera robots, which are unicycle mobile robots. Their kinematic model can be described by the well-known equations (cf. Equation 2).

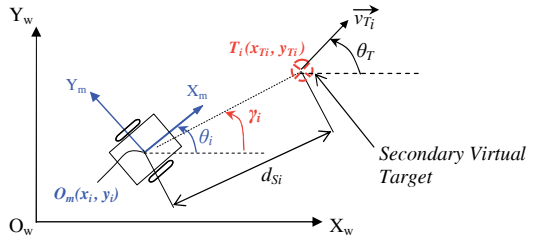


Fig. 3. Attraction to a dynamic target.

$$\begin{cases} \dot{x}_i = v_i.cos(\theta_i) \\ \dot{y}_i = v_i.sin(\theta_i) \\ \dot{\theta}_i = \omega_i \end{cases} \quad (2)$$

where θ_i, v_i and ω_i are respectively the robot orientation, the linear and angular velocities.

The set-point angle that the robot must follow, to reach its dynamic target, is given by

$$\theta_{S_{ati}} = \arcsin(b \sin(\theta_T - \gamma_i)) + \gamma_i \quad (3)$$

Where $b = \frac{v_{T_i}}{v_i}$. γ_i is the angle that the robot would have if it was directed to its target (cf. Figure 3). This set-point has been obtained by keeping γ_i constant. More details and proofs are available in [11].

The corresponding set-points (P_{S_i}, θ_{S_i}) (cf. Figure 1) given by the *Attraction to Dynamic Target* controller are composed by:

- $(P_{S_i} = (x_{T_i}, y_{T_i}))$: the current position of the dynamic target (cf. Figure 3),
- $(\theta_{S_i} = \theta_{S_{ati}})$ given by equation (3).

C. Obstacle Avoidance controller

A particular attention is given to this task since the objective of the paper is to extend the already proposed orbital obstacle avoidance strategy [18], so that it becomes more appropriate to deal with dynamic obstacles. As cited in section I, common potential field approaches for obstacle avoidance are not used because of their drawbacks in robots formation. The task is then performed through the limit cycle methods. The robot follows the limit cycle vector fields described by the following differential equations:

$$\begin{aligned} \dot{x}_s &= (sign) y_s + x_s(R_I^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= -(sign) x_s + y_s(R_I^2 - x_s^2 - y_s^2) \end{aligned} \quad (4)$$

where $(x_s,\ y_s)$ corresponds to the relative position of the robot according to the center of the convergence circle (characterized by an R_I radius).

The function $sign$ allows to define the direction of the trajectories described by these equations. Hence, two cases are possible

- $sign = 1$, the motion is clockwise.
- $sign = -1$, the motion is counterclockwise.

Figure 4 shows the limit cycles with a radius $R_I = 1$. Obstacles are then modeled as circles of R_I radius. The latter is chosen as the sum of the obstacle radius, the robot radius and a safety margin.

The set-point angle $\theta_{S_{oa}}$ of the Obstacle Avoidance controller is given by the the following relation

$$\theta_{S_{oa}} = \arctan(\frac{\dot{y}_s}{\dot{x}_s}) \quad (5)$$

The corresponding set-points (P_{S_i}, θ_{S_i}), when the *Obstacle Avoidance* controller is chosen by *Hierarchical Set-Point Selection block* (cf. Figure 1), are defined such that $P_{S_{oa}}$ corresponds to the center position of the obstacle (x_o, y_o) whereas $\theta_{S_i} = \theta_{S_{oa}}$.

It is noticed that previous works on limit-cycle methods applied to obstacle avoidance [17], [18] do not consider dynamic obstacles. Here, it is proposed to extend this reactive method to deal with them.

According to the nature of the obstacle, three cases are considered:

1) static obstacles,
2) dynamic obstacles,
3) robots of the same system.

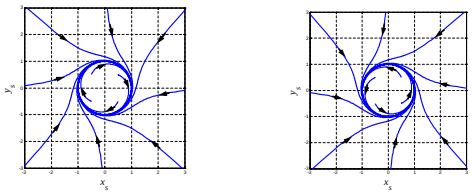
These strategies are explained in the next paragraphs.

1) Static obstacles: The same strategy proposed in [18] is maintained. Summarily, the value of $sign$ is specified by the ordinate of the robot y_s in the relative obstacle's frame $(O_o X_o Y_o)$ (cf. Figure 5). The X_o axis of this orthonormal frame is defined thanks to two points: the center of the obstacle (which makes the origin of the frame) and the target to reach.

$$\text{sign} = \begin{cases} 1 & \text{if } y_s \geq 0 \text{ (clockwise avoidance)} \\ -1 & \text{if } y_s < 0 \text{ (counterclockwise avoidance)} \end{cases} \quad (6)$$

Figure 5 shows an example of a robot choosing its avoidance direction (clockwise) thanks to its relative ordinate $y_s > 0$. The chosen direction by this strategy allows then to join the target by the side offering the smallest covered distance.

2) Dynamic obstacles: When a movement of the obstacle position is detected, it is considered as a dynamic obstacle by the robot. The objective for the robot is always to choose the most suitable side of avoidance (clockwise or counterclockwise) which allows to succeed this mission. The proposed solution is always to act on the function $sign$ (cf. Equation 4). Nevertheless, for dynamic obstacles, the ordinate y_s cannot be used as the adequate information to



(a) Clockwise $(sign = 1)$ (b) Counter-Clockwise $(sign = -1)$

Fig. 4. Possible trajectories of the limit-cycles

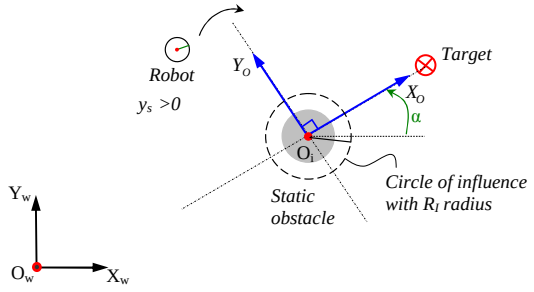


Fig. 5. Avoiding a static obstacle.

decide on the avoidance direction. In figure 6, it can be noticed that if the robot decides a clockwise motion (based on its relative positive ordinate $y_s > 0$), it fails to avoid this obstacle. In fact, the robot will go in the same direction as the obstacle (vector $\vec{v}_O$ on the figure). It may then uselessly diverge from its target by persisting in this direction.

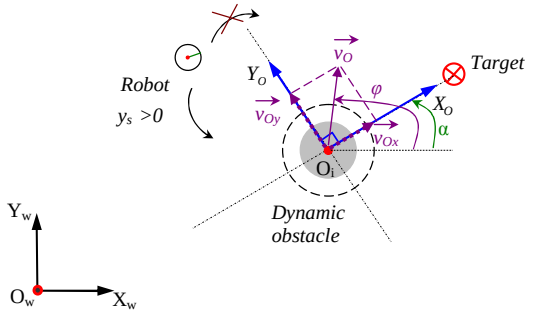


Fig. 6. Avoiding a dynamic obstacle.

Rather than analyzing y_s, it is then proposed that the robot uses the obstacle's vector velocity $\vec{v}_O$. The idea is to project this vector on the Y_o axis of the relative frame $(O_o X_o Y_o)$ defined in paragraph II-C.1. Noted v_{O_y}, this projection is expressed as

$$v_{O_y} = v_O sin(\varphi - \alpha) \tag{7}$$

where α and φ define the direction of the X_o axis and $\vec{v}_O$ in the absolute frame respectively. The function $sign$ (cf. Equation 4) is then defined according to v_{O_y} as follows:

$$\text{sign} = \begin{cases} 1 & \text{if } v_{O_y} \leq 0 \text{ (clockwise avoidance)} \\ -1 & \text{if } v_{O_y} > 0 \text{ (counterclockwise avoidance)} \end{cases} \tag{8}$$

By using the projection v_{O_y} of the obstacle velocity, the obstacle is always avoided round the back such that the robot does not cut off the obstacle's trajectory.

3) Robots of the same system: One can consider that every robot of the MRS is treated as a dynamic obstacle and projects its velocity vector to deduce the side of avoidance (cf. Equation 8). However, a conflict problem could appear when, for instance, two robots have to avoid each other. Once each robot projects the velocity vector of the other one, they have opposite directions of motion. They can endlessly hinder each other which leads to divergence from their targets. This problem is illustrated by a simulation example in figure 7.

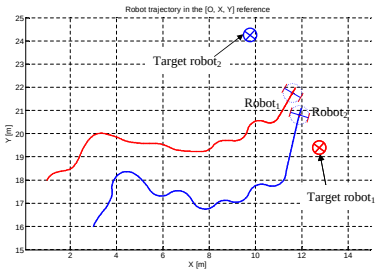


Fig. 7. Divergence of the robots from their targets due to two different directions of avoidance.

To deal with this kind of conflicts, and assuming that each robot is able to identify those of the same system, it is proposed to impose one reference direction for all the system. Hence, when one robot detects a disturbing robot of the same group, it avoids it counterclockwise.

D. The control law block

This block allows for the robot i to converge to its set-point given by the *Hierarchical set-point selection* block(cf. Figure 1). It is expressed as

$$v_i = v_{max} - (v_{max} - v_T)e^{-(d_{S_i}^2/\sigma^2)} \tag{9a}$$

$$\omega_i = \omega_{S_i} + k_1 \tilde{\theta}_i \tag{9b}$$

where

- v_{max} is the maximum linear speed of the robot,
- σ, k_1 are positive constants,
- v_i and ω_i are linear and angular velocities of the robot. $w_{S_i} = \dot{\theta}_{S_i}$.

$$\tilde{\theta}_i = \theta_{S_i} - \theta_i \tag{10}$$

where θ_{S_i} is the set-point angle according to the active controller and was already computed (cf. Equation (3), (5)). By derivating

$$\dot{\tilde{\theta}}_i = w_{S_i} - \omega_i \tag{11}$$

Consider the well known Lyapunov function

$$\text{V} = \tfrac{1}{2}\tilde{\theta}_i^{\,2} \tag{12}$$

The angular control law is asymptotically stable if $\dot{V} < 0$.

$$\dot{V} = k_1 \tilde{\theta}_i \dot{\tilde{\theta}}_i$$

By replacing equation (11) in the control law (9b), we get $\dot{\theta}_i = -k_1 \tilde{\theta}_i$

and $\dot{V}$ becomes $\dot{V} = -k_1 \tilde{\theta}_i^2 < 0$

for every $\tilde{\theta}_i \neq 0$ since $k_1 > 0$.

In addition to the obstacle avoidance controller (cf. Section II-C), it is proposed to better prevent the collision risk. The idea, described in next section, is to increase time of maneuvering for the robot by reducing the relative velocity between it and the hindering obstacles.

III. TOWARD A NULL RISK OF COLLISION

It is here proposed to modify the linear velocity of the robot according to the distance separating it from the hindering obstacle through a function ψ. This function is called a *penalty function*. The linear velocity of each robot i (cf. Equation 9a) is then modified by the penalty function related to the obstacle j and noted $\psi_j(d_{ij})$. d_{ij} is the distance separating robot i and obstacle j.

To define $\psi_j(d_{ij})$, the robot i is surrounded with two additional virtual circles (cf. Figure 8):

- a circle of radius R_{ext} such that $R_{ext} \geq R_{Ii}$, (R_{Ii} is the radius of the limit cycle surrounding the robot),
- a circle of radius R_{inti} such that $R_{inti} < R_{Ii}$.

The penalty function can then be defined as follows:

$$\psi_{\mathrm{j}}(d_{ij}) = \begin{cases} \frac{(d_{ij}-R_{int_i})}{(R_{ext}-R_{int_i})} & (R_{int_i} < d_{ij} < R_{ext} \text{ and } x_{Oj/Ri} > 0) \\ 0 & d_{ij} \leq R_{int_i} \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

where $x_{Oj/Ri}$ is the relative position of the obstacle j in the relative frame of the robot i (cf. Figure 8). In fact, by imposing $x_{Oj/Ri} > 0$, only the obstacles in front of the robot i impact its velocity. Robots behind it do not modify it.

When the robot is hindered by M obstacles, its new velocity noted $v_i^{'}$ is then given by

$$v_i^{'} = v_i \prod_{j=1, j\neq i}^{M} \psi_j(d_{ij}) \tag{14}$$

where v_i is the velocity of the robots given by the *Control Law* block without penalty (cf. Equation 9a).

Note that if one hindering obstacle is a robot of the same MRS, the penalty function may cause local minima where two robots (at least) are stopped by each other. In fact, if $R_{int_i} = R_{int_j}$ and $d_{ij} \leq R_{int_i}$, then $\psi_j(d_{ij} = \psi_i(d_{ji} = 0$. This means that $v_i^{'} = v_j^{'} = 0$ (cf. Equation 14). To overcome this minima, and for every couple of robots k and l such that $(k, l \in \{1..N\})$, radius R_{intk} and R_{intl}, are attributed such that

$$|R_{intk} - R_{intl}| \geq \xi$$

where ξ is the tolerance margin of the robots sensor.

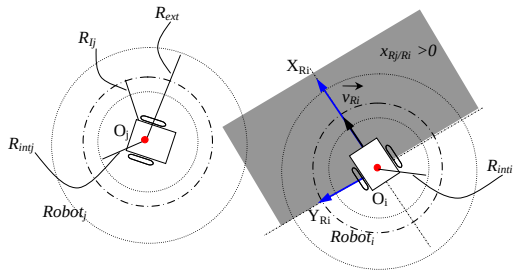


Fig. 8. Virtual circles defining the penalty function $\psi_j(d_{ij})$.

IV. EXPERIMENTAL RESULTS

Experimentations are made on Khepera III robots. A central camera, at the top of the platform gives positions of all the robots and the obstacles thanks to circular bar codes installed on them. The objective on the long view is to use the local sensors of the robots in order to get a completely decentralized architecture. Experimental results can be illustrated in two paragraph : first, the dynamic obstacle avoidance is shown thanks to a robot joining a static target. In the second paragraph, three robots avoid each other before attaining a dynamic virtual structure.

A. Avoiding a dynamic obstacle

One robot has to reach its static target $v_T = 0$ (cf. Equation 9a) while avoiding an other robot considered as a dynamic obstacle. The strategy of avoiding dynamic obstacles using the projection of their velocity vector is then shown. Figure 9 shows the robot and the obstacle trajectories. It can be seen that the robot avoids the obstacle by surrounding it behind and attains its final target. Figure 10 shows the variation of the linear velocity of the robot and the distance separating it from the obstacle. It can be seen that when this distance is $d \leq R_{ext}$, the robot decelerates (its velocity is decreasing) modified by the penalty function ψ (cf. Equation 13) of the obstacle. When the robot avoids it, it accelerates again and starts deceleration by reaching the target (cf. Equation 9a).

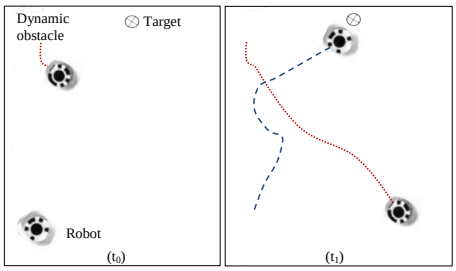


Fig. 9. Avoiding a dynamic obstacle.

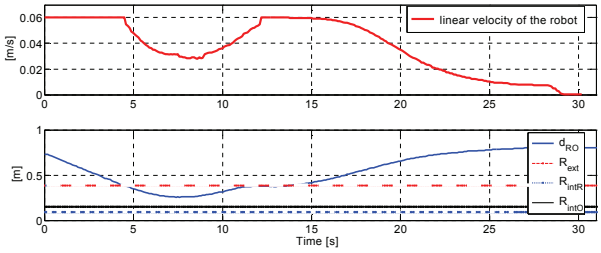


Fig. 10. Linear velocity of the robot and distance d_{RO} separating the robot from the obstacle.

B. Attaining a formation while avoiding collision between the robots

Three robots have to join a triangular virtual structure. They are put in an initial condition such that they must avoid each other using the proposed obstacle avoidance controller (robots of the same system) (cf. Section II-C.3). It is observed that the robots proceed to collision avoidance

before attaining the formation. No conflict was observed since avoidance is done in one direction (counterclockwise). The formation is successfully attained as shown in figure 11 illustrating the trajectories of the three robots. Moreover, the penalty function allows to each one deceleration when it approaches other robots offering a bigger time of maneuvering. Figures 12 and 13 represents the variation of the linear velocities and the distances separating each other respectively. By analyzing them, it can be seen how the penalty functions appear when the distances become small $(d_{ij} \leq R_{ext})$. This explains the diminution of the velocities before reaching the target (cf. Figure 12).

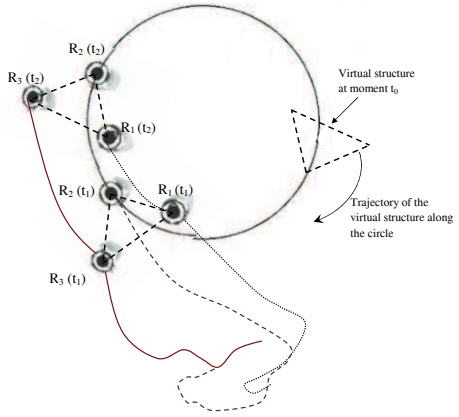


Fig. 11. Trajectories of the robots attaining the formation.

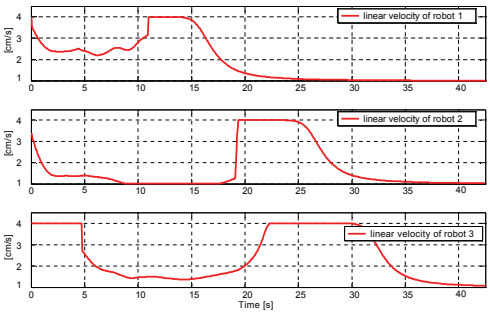


Fig. 12. Linear velocities of the robots.

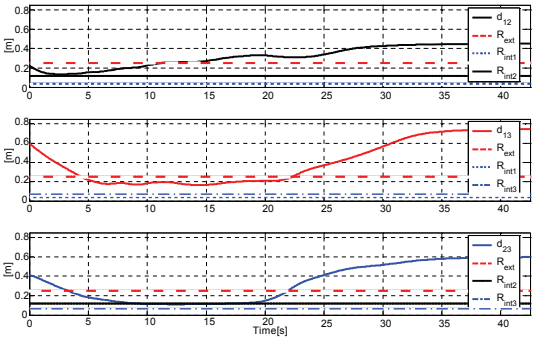


Fig. 13. Distances d_{ij} between the robots.

V. Conclusion

A new reactive collision avoidance method, based on limit-cycle approach, is proposed to deal with multi-robot system (MRS). Hence, the control architecture, which allows the navigation in formation of a MRS, is enriched with a more flexible and reliable obstacle avoidance strategies. This allows to deal with static and dynamic obstacles and permits also to avoid collisions between the robots of the same group. Thus, some conflicts, which were possible when using limit-cycle method for dynamic obstacles, are solved. In addition, the proposed *penalty function* makes the obstacle avoidance controller more robust against collisions, since it permits to take into account the different local interactions between robots and their environment. Future works will consider the kinematic constraints of the robot while generating the convergence toward the control set-points. The objective is to insure the safety of the robot and the control feasibility.

References

[1] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17:760–772, 1998.

[2] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH Eurographics symposium on computer animation*, 2009.

[3] A. Pongpunwattana and R. Rysdyk. Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. *Journal of Aerospace Computing, Information and Communication*, 1:580–604, 2004.

[4] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5:90–99, 1986.

[5] E. Lalish, K.A. Morgansen, and T. Tsukamaki. Formation tracking control using virtual structures and deconfliction. *IEEE Conference on Decision and Control*, 2006.

[6] P. Ogren, E. Fiorelli, and Leonard N. E. Formations with a mission: Stable coordination of vehicle group maneuvers. In *15th International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[7] S. Mastellone, D.M. Stipanovic, and M.W. Spong. Remote formation control and collision avoidance for multi-agent nonholonomic systems. In *IEEE International Conference on Robotics and Automation*, pages 1062–1067, 2007.

[8] R. Zapata and P. Lepinay. Reactive behaviors of fast mobile robots. *Journal of Robotics Systems*, 11:13–20, 1994.

[9] P. Fraisse, R. Zapata, W. Zarrad, and D. Andreu. Remote decentralized control strategy for cooperative mobile robots. *Advanced Robotics Journal, Robotics Society of Japan*, 19:1027–1040, 2005.

[10] E. Lalish and K.A. Morgansen. Distributed reactive collision avoidance. *Autonomous Robots*, 32, 2012.

[11] A. Benzerrouk, L. Adouane, L. Lequievre, and P. Martinet. Navigation of multi-robot formation in unstructured environment using dynamical virtual structures. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[12] X. Li, J. Xiao, and Z. Cai. Backstepping based multiple mobile robots formation control. In *IEEE International Conference on Intelligent Robots and Systems*, pages 887 – 892, 2005.

[13] K. D. Do. Formation tracking control of unicycle-type mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 527–538, 2007.

[14] T. Balch and R.C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Autmation*, 1999.

[15] G. Antonelli, F. Arrichiello, and S. Chiaverini. The nsb control: a behavior-based approach for multi-robot systems. *PALADYN Journal of Behavioral Robotics*, 1:48–56, 2010.

[16] H.K. Khalil. *Frequency domain analysis of feedback systems (chapter 7)*. 2002.

[17] D. Kim and J. Kim. A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. *Robotics and Autonomous Systems*, 42:17–30, 2003.

[18] L. Adouane. Orbital obstacle avoidance algorithm for reliable and on-line mobile robot navigation. In *9th Conference on Autonomous Robot Systems and Competitions*, May 2009.

[19] J.O. Kim, C.J. Im, H. Shin, K. Y. Yi, and H. G. Lee. A new task-based control architecture for personal robots. *International Conference on Intelligent Robots and Systems*, 2:1481–1486, 2003.

[20] L. Adouane, A. Benzerrouk, and P. Martinet. Mobile robot navigation in cluttered environment using reactive elliptic trajectories. In *18th IFAC World Congress*, 2011.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session III

Interactive session

- **Title: Development of an Autonomous Vehicle for High-speed Navigation and Obstacle Avoidance**
  Authors: J.H. Ryu, D. Ogay, S. Bulavintsev, H. Kim, and J.S.Park

- **Title: Kinodynamic motion planning with state Lattice Motion Primitives**
  Authors: M. Pivoraiko and A. Kelly

- **Title: Detection of Moving and Stationary Objects at High Velocities using Cost-Efficient Sensors, Curve-Fitting and Neural Networks**
  Authors: F. Mirus, J. Pfadt, C. Connette, B. Ewert, D. Grudl, A. Verl

- **Title: ESTRO: Design and Development of Intelligent Autonomous Vehicle for Shuttle Service in the ETRI**
  Authors: J. Byun, K.I. Na, M. Noh and S. Kim

- **Title: An effective 6DoF motion model for 3D-6DoF Monte Carlo Localization**
  Authors: A. L. Ballardini, A. Furlan, A. Galbiati, M. Matteucci, F. Sacchi, D. G. Sorrenti

- **Title: Visual trajectory learning and following in unknown routes for autonomous navigation**
  Authors: D. A. Marquez-Gamez and M. Devy

- **Title: Eigen analysis and gray alignment for shadow detection applied to urban scene images**
  Authors: T. Souza, L. Schnitman and L. Oliveira

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Development of an Autonomous Vehicle for High-speed Navigation and Obstacle Avoidance

Jee-Hwan Ryu, *Member, IEEE,* Dmitriy Ogay, Sergey Bulavintsev,
Hyuk Kim, and Jang-Sik Park

***Abstract*— This paper introduces the autonomous vehicle Pharos, which participated in the 2010 Autonomous Vehicle Competition organized by Hyundai-Kia motors. Pharos was developed for high-speed on/off-road unmanned driving avoiding diverse patterns of obstacles. For the high speed traveling up to 60 Km/h, long range terrain perception, real-time path planning and high speed vehicle motion control algorithms are developed. This paper describes the major hardware and software components of our vehicle.**

I. Introduction

The first autonomous vehicle competition in South Korea organized by Hyundai-Kia Motors took place on November, 2011 [11]. The mission of the competition was unmanned traveling about 4 Km on/off road clearing 7 different patterns of obstacles. Lane keeping and stop within 1 m of crosswalk were also included. Eleven universities in South Korea were participated in the main competition out of 20 initial applicants, and "Pharos", developed by our team, finished the course in 8 min 52 sec clearing all the missions except crosswalk, which got 5 min penalty, and Pharos took 4^{th} place.

Recently there have been many research activities in autonomous vehicle area. Especially, DARPA Grand Challenge [1], [8] and Urban Challenge [10] made a big progress on the area of autonomous vehicle. However, there are still many open issues for high speed unmanned traveling such as long range terrain perception [9], real-time obstacle avoidance and trajectory planning [3], [6], and high speed vehicle motion control [5] et al.

In this paper, autonomous vehicle based on real Sport Utility Vehicle (SUV) is introduced motivated by the competition. The main challenging issue in the development of the vehicle was to build a reliable system, able to traveling high speed up to 60 Km/h through on-road and unstructured off-road while avoiding different types of obstacles. To satisfy these requirements, new methods were developed and extended based on existing methods in the field of autonomous navigation such as long-range obstacle detection and mapping, real-time collision avoidance and trajectory planning, and stable vehicle control on slippery and rugged terrain.

This work was supported partly by the R&D program of the Korea Ministry of Knowledge and Economy (MKE) and the Korea Institute for Advancement of Technology (KIAT). (Project: 3D Perception and Robot Navigation Technology for Unstructured Environments, M002300090)

Jee-Hwan Ryu is with the School of Mechanical Engineering, Korea University of Technology and Education, Cheonan, South Korea jhryu@kut.ac.kr

Dmitriy Ogay is with the School of Computer Science and Engineering, Korea University of Technology and Education, Cheonan, South Korea cactus@ kut.ac.kr

Sergey Bulavintsev, Hyuk Kim, and Jang-Sik Park are with the School of of Mechanical Engineering, Korea University of Technology and Education, Cheonan, South Korea {sergey,perseus,ganggai}@kut.ac.kr

II. Hardware Design

Fig. 1 shows the outlook of Pharos at the competition. Pharos is based on a Gasoline-powered Hyundai Santafe CM. Reinforced front bumper has been installed to protect the vehicle from the environmental impact.

Fig. 1. Pharos was standing at the finishing line on the competition track

A geared DC motor is attached to the steering column to allow electronic steering control. Required torque and rotational speed was estimated through the experimental analysis. We found that 4.41 Nm continuous torque with 3000 rpm was required for controlling the steering column. Smart motor from Animatic Co. with 1:4 gear ratio was selected as a steering motor. With this motor, 5.8 Nm continuous torque with 4000 rpm can be achieved at 24V, which is sufficient to control the steering column.

Fig. 2 shows the assembled 3D CAD model of the steering actuation mechanism. one-to-one pulley powered transmission mechanism was used, and steering angle sensor was moved to the motor output axis to measure the absolute steering angle.

To motorize braking system, another geared DC motor is installed near by the braking pedal. Through the initial experiment, we found that 60 mm travel distance with 8 N pressing force is required max. for controlling the breaking

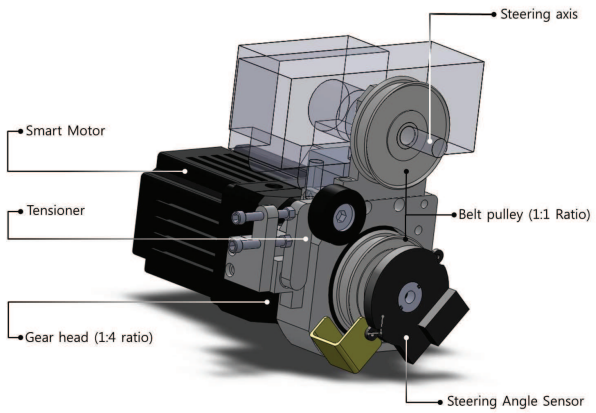


Fig. 2. Assembled 3D CAD model of the steering actuation mechanism

system. one-to-five gear ratio was used to give enough force. It also allows fine position control of the breaking pedal.

Throttle is also modified to be controlled electronically through APS signal modification. APS signal is modified to control the throttle. From 0.8 V to 5 V was given for idling to maximum acceleration.

Vehicle data, such as steering angle, vehicle speed, APS signal and et al. are automatically sensed and communicated to the computer system through Ethernet interface.

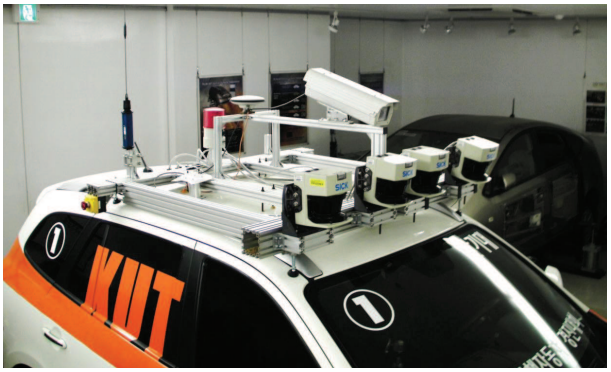


Fig. 3. View of custom made roof rack with sensors

Fig. 3 shows the custom-made roof rack. Most of the sensors are installed on it. Four SICK laser scanners are installed on the roof rack since it can provide best visibility of the terrain. All scanners are facing forward along the driving direction of the vehicle, but with slightly different angles. A single CMOS camera with housing is also installed on the roof rack for crosswalk detection. GPS receiver, RF modem and a radio antenna for E-stop are also installed on the roof rack. The E-stop system is provided by Hyundai-Kia motors for allowing the chasing vehicle safely stop the vehicle in emergent situation with wireless link. Three manual E-stop buttons are installed also on the roof rack and backside of trunk.

Pharos's controllers and communication system are located inside of trunk as shown in Fig. 4. Six Compact PCI, NI-CompactRIO, Gigabit Ethernet switches and various

Fig. 4. Controllers and communication system in the trunk of the vehicle

types of interfaces for physical sensors and actuators are installed on a shock-mounted rack. Custom-made power system with backup batteries are installed underneath of the rack for supplying power to all the electrical components. A six degree-of-freedom IMU is rigidly attached to the vehicle frame underneath the vehicle roof.

The added instrument is required approximately 2.4 KW in addition. Two 12V alternators are installed in addition to supply power, and it provides 24V, 3.4 KW. Therefore, Pharos can even run all the system more than an hour without recharging.

III. Software Architecture

A. Overview

The main principles that we wanted to follow when developing the system architecture were: reliability of a system, ability to distribute software over several independent computers, easy configuration, simulation and development. To reach these requirements we have chosen an architecture, where each module was an independent program that could be run stand alone in a separate process of an operating system. Communication between modules is done through publish-subscribe mechanism.

There is no central process, which encapsulates modules, as it could reduce reliability of the system, if such a process dies. All communication is done through a lightweight messaging server, which supports connectivity through network protocols. It allowed us to easily distribute modules over several computers, and even gave opportunity to run different modules on different operating systems.

This approach also makes development of the modules simpler, because each module is an independent program and can be run stand alone in an environment, where all inputs and outputs can stubbed. For example we could substitute actual data from the sensors by data from file for simulation.

In a running integrated system all blocks are put on watchdogs, so they are restarted if fail, or they can be controlled remotely from the user interface or health monitoring system.

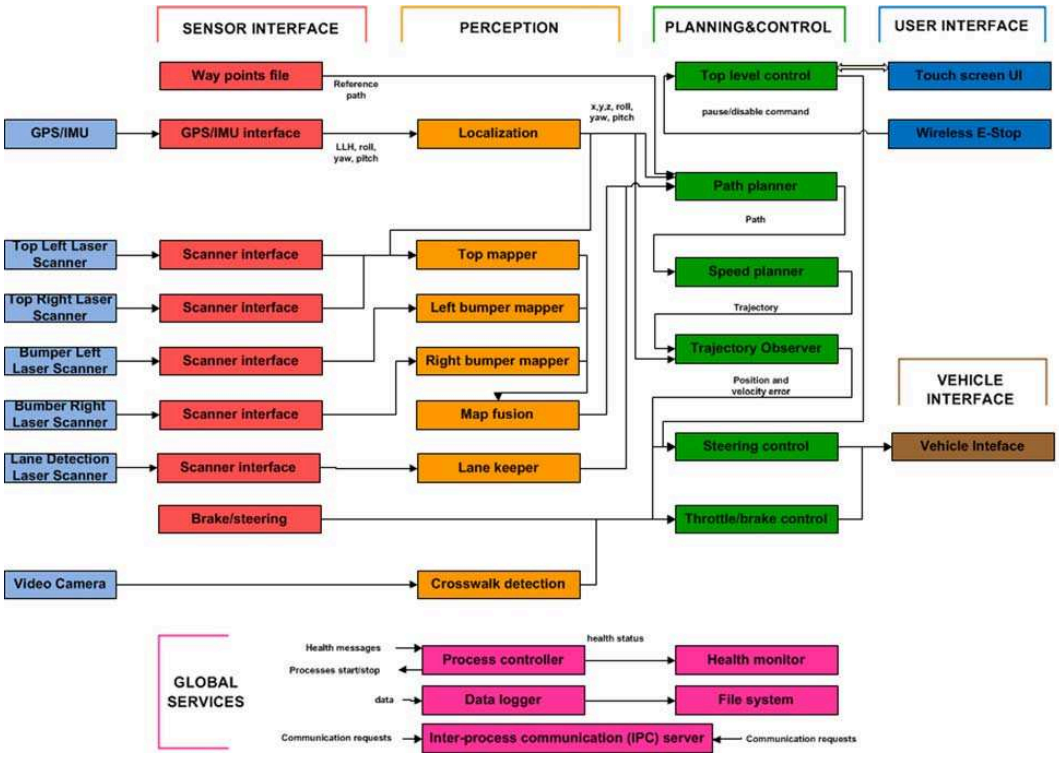


Fig. 5. Overview of the software architecture

B. System Components

Our system has multiple processing layers, when data flow from sensors through consecutive layers, until it reaches steering and braking-acceleration actuators. These layers are: sensor interface, perception layer, planning layer, and control layer. Each layer consists of several modules as shown in Fig. 5.

Sensor interface layer consists of modules that receive data directly from sensors, using sensor protocols, and then publishes the data in a format used inside our system. They run on frequency determined by corresponding sensors, which are 75Hz for laser scanners and 100 Hz for GPS/IMU.

Perception layer has several independent parts that are: localization for estimating vehicle state, laser mapping for generating obstacle occupancy grid map from laser scanner data, and vision mapping for detecting road markings. Data to the path planner are sent at 10 Hz frequency.

Planning layer consists of path planning module that uses 2D obstacle map to generate traversable path, speed planning layer that plans speed taking into account current vehicle speed, speed limits imposed by competition requirements, and shape of the generated path.

Control layer consists of a Trajectory Observer, which calculates difference between planned path and current vehicle position and sends it to the motion control module, which is run in a separate real-time system. Unlike path planner, which is non-deterministic, Trajectory Observer is run at 100 Hz frequency, monitors vehicle position and stops vehicle if no data from path planner come, which can occur in case of a process failure.

All modules also generate health monitoring messages, which are listened by a health monitoring block.

IV. Obstacle Detection and Mapping

To make safely avoid obstacles, Pharos must be able to create a precise terrain map with sufficient range of view for undertaking appropriate actions. The map should contains accurate position of the obstacles to maintain the movement of the vehicle even in the narrow path, comparable with the width of the vehicle. Detection range of the obstacles is considered as well since that faster speed of the vehicle results in the longer perceiving distance. To satisfy these main criteria, medium range laser scanners were chosen. It has 75 *Hz* update rate with accuracy 1cm on the 30 m distance. Laser scanner system of the Pharos consists of three laser scanners, mounted on the roof, tilted downward and two scanners, installed on the front bumper with 45 degree yaw angle with respect to the vehicle to provide wider area of view. Different obstacle classification algorithms are used for scanners installed on the roof and on the bumper. Each laser scanner acquires distance information in its own local coordinate frame. By using estimated position and orientation of the vehicle, data from scanners are transformed into points in global coordinate frame and represent 3D point cloud. Each point in 3D point cloud can be described as (X_m^i, Y_m^i, Z_m^i) where m is the index of the individual scanner and i is the index of each measurement in one scan.

Deterministic algorithm is used for the roof laser scanners because it has shown more reliable performance compared

to other algorithms that had been tested. The classification method is based on the occupancy grid map [2]. The map represents 2D map consists of cells, and each cell contains position information and one of the possible conditions: occupied or unknown. The size of the cells is decided considering maximum resolution at the distance 30 m. To classify a cell as obstacle, two nearby points in 3D point cloud must be found and their vertical distance must satisfy condition $|Z_k^i - Z_l^j| > \varepsilon$, where i and j are indices of two closest points in 3D cloud point from two different laser scanners and ε is the threshold, assumed to being obstacle. If no such points can be found or no value is acquired, this cell is assigned as unknown. Unknown zone considered as drivable. Fig. 6 shows the occupancy grid map, drawn in our campus. Gray color means unknown area or drivable and white color represents obstacles.

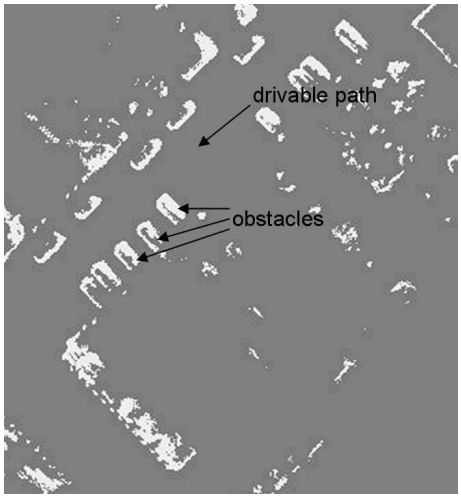


Fig. 6. An example of occupancy grid map

The advantage of this algorithm is that the measurements from two different scanners are used to detect obstacles. This excludes possibility to detect fake obstacles due to the error in the orientation estimation of the vehicle. In addition, it is possible to make six combinations when three scanners are used, therefore reliability of the algorithm can be increased. On other hand, algorithm can be possibly less reliable in case of fault in laser scanner hardware. Bumper laser scanners use different way for obstacle classification. Those are mounted on some known height relate to the ground and scans in horizontal plane. Cell is assigned to be an obstacle if points with same X and Y location satisfies condition $\sum_{j=1}^{n} p_j(Z_j > |h-\delta|) \cdot w > 0.5$, where j is the time index for the series of height measurements acquired for same cell, w is weight of the possibility that point is obstacle and δ is vertical distance that determines size of the obstacle. If same point has been classified as obstacle several times, likelihood of the obstacle presence in this location increases and obstacle is detected.

In practice both algorithms have shown reliable performance and allow our vehicle avoiding obstacles on the distance around 30 m ahead with speed up to 60 km/h.

V. Road Boundary Estimation

One of the problems that can occur during autonomous driving is reference path drifting, due to the shifting of the base coordinates of the DGPS, which varies depend on the various conditions. To avoid it, we develop algorithm, which helps to determine the position of the vehicle with respect to the road boundary and removes the effect of the drifting. On the asphalt road, where static obstacles usually aren't encountered, vehicle has to keep traveling right side of the road. Therefore our task simply was to define lateral distance between car and one of the road sides. To distinguish actual road from off-road, we use reflection properties of the road. Flat asphalt surfaces have lower reflection value than rough terrain. Based on this reflection, occupancy grid can be built. The cell is assumed as non-drivable if reflection value at that location higher than certain limit. Accumulated average reflection value is used. The reference path assumed to be parallel to the road side, but usually lateral offset changes over the time and it involves the unstable behavior of the vehicle when unfiltered offset is used to correct car position. To find lateral distance, one-dimensional low-pass Kalman filter is used. The state of the Kalman filter is distance between reference path and the road boundary. The Kalman filter searches for largest offset along discrete search pattern orthogonal to the reference path. The Fig. 7 shows an example of discrete search pattern. In that case the road boundaries change slowly. Sudden increases in reflection, for example when obstacles appear which normally have higher reflection value, result in small affect in the boundary road estimation. Output from Kalman filter defines offset which is used by path planner, which necessary to prevent road boundary crossing and removing influence of the DGPS drift.

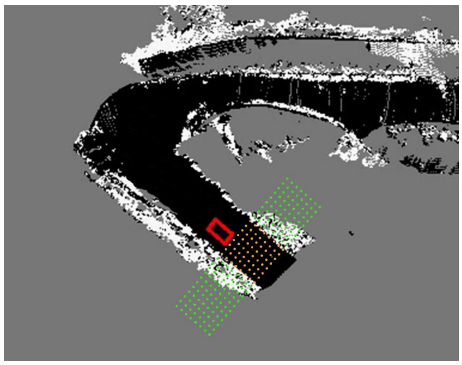


Fig. 7. An example of road boundary estimation

VI. Path and Speed Planning

A. Mission

The main mission that car should accomplish is the following of a pre-recorded reference path. Artificial and natural obstacles can be present on a reference path, so vehicle should avoid them, and also edges of a road can be natural obstacle, because of GPS localization drifting.

The reference path is stored in a text file. Before being given as an input to the path planner, the reference path

is smoothed, and such parameters as tangential vector and curvature of the path are calculated for each reference point.

B. Path Planner

For our implementation, we have chosen RRT [7] based approach, as it can drive vehicle through a very narrow path and can take kinematic constraints of a vehicle into account. But computation speed is a challenge for the development of real-time algorithms [4]. We wanted our vehicle to response faster to the environment changes, because the range of car's perception was limited. And fast path planner response would allow us to increase car's speed.

To cope with those challenges we made some modifications to original RRT algorithm and also developed a new method to increase computing time efficiency of the path planner. It should be mentioned that our path planner is run in a single thread and has re-planning time about 100 ms, which is enough to drive vehicle at speeds up to 60 km/h, given car's perception range.

Original RRT algorithm is modified in several ways, including biased sampling, which is made along lines perpendicular to the reference way points. The state consists of three components, which are: x and y coordinates, and a heading angle of a vehicle.

To plan kinematic behavior of a vehicle we assumed that vehicle travels in short arcs. This let us use curvature of a planned path to calculate lateral acceleration and determine maximum speed, given limited lateral acceleration, which is the safe speed. Then we use time component based on this safe speed in our cost function, which measures distance between states in RRT. The main optimization criteria are fast and smooth motion. Fig. 8 shows an example of the path planning inside of an artificially made narrow tunnel. The area beyond the walls on Fig. 8(a) is valid for sampling in original RRT algorithm, but that area is not reachable by a car. Fig. 8(b) shows that the proposed planner allows us to achieve computing time efficiency by not propagating path to the points that are potentially not reachable.

Before planned path is sent as output it is run through a Savitzky-Golay low-pass filter at first to be smoothed, but mainly to calculate tangential vectors and curvatures of the path at every point.

C. Speed Planner

The main task for a speed planner is to follow maximum safe speed, while taking into account speed limit zones, that are recorded together with reference path, and maximum braking deceleration to avoid collisions if there is some sudden change in a safe speed. As it was mentioned before, safe speed is determined by a curvature of a planned path. And, if there is a sudden change in a path curvature, speed planner should react in advance.

D. Trajectory Observer

Trajectory Observer is a special block, running at a high frequency which measures difference between current state of a vehicle and planned path. Then it sends the data to the control block. There were two reasons to introduce this block. At first it is an interface between the whole system run on a general purpose operating system, which is linux in our case and a real time system. At second it improves safety of a vehicle, as it is deterministic and does have delays or failures like path planner.

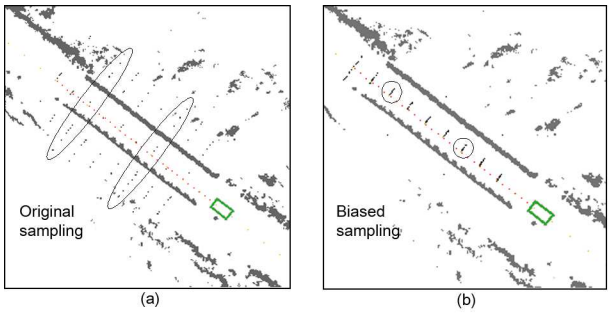


Fig. 8. Comparison of sampling distribution (black dots) of non-optimized algorithm (a), and optimized algorithm (b). Passing artificially made narrow tunnel

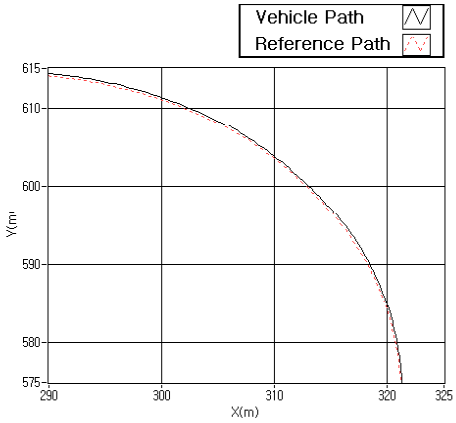
VII. Real-Time Vehicle Motion Control

Once the errors to the reference trajectory of the vehicle is given from the trajectory observer, the motion controller produces appropriate steering, throttle and brake command to achieve the given trajectory. This issue will be described in two parts: steering control and speed control.

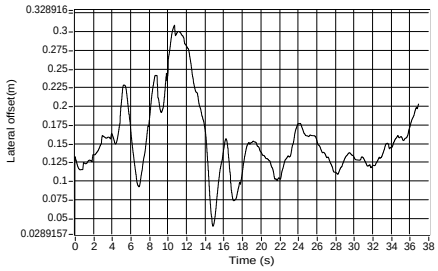
A. Steering Control

Lateral offset, angle difference and curvature of the reference path are given to the steering controller as inputs. The steering controller gives steering commands to the steering motor at a rate of 100 Hz. The basic steering angle control law is very similar to the one in Stanley [8] except look ahead consideration. Simple kinematic based PD feedback controller is used for compensating Lateral offset, which measures the lateral distance of the center of the vehicle's front wheels from the nearest point on the reference trajectory, and angle difference, which is the orientation of the nearest path segment, measured relative to the vehicle's own orientation. Basically, compensating angle difference only allows our vehicle to track the reference trajectory at some level, however without lateral offset consideration it can not compensate steady state error. To copy with the continuous change of the reference trajectory, curvature of the trajectory at an adaptive look ahead point was used as a feed forward control input. Each gains were scheduled to the vehicle velocity, and has proven stable trajectory following on terrain from pavement to off-road, and trajectories with tight curvature.

Fig. 9 shows steering performance while Pharos was making a corner of 40 m radius curved road. Steering controller maintained within 30 cm lateral offset around 60 km/h.



(a) Tracking performance of the steering controller



(b) Lateral offset error of the vehicle

Fig. 9. Steering performance of the vehicle on 40m radius curved road

B. Speed Control

Speed planner and cross walk detection module give speed command to a low-level speed controller. The low-level speed controller translates the speed commands from each modules into actual throttle and brake commands. The minimum of the two recommended speed is always used.

Once the desired speed is determined, low-level speed controller control the brake level and throttle level exclusively. It treats the break and throttle as two opposing single-acting actuators that produce longitudinal force on the vehicle. The controller computes weighted sum plus weighted integration of the speed error. Based on the sign of the computed value, either brake or throttle level is controlled. By considering dead-zone, the controller is able to avoid the chattering behavior. Fig. 10 shows the control performance of the proposed speed controller. It tracks the desired speed very well.

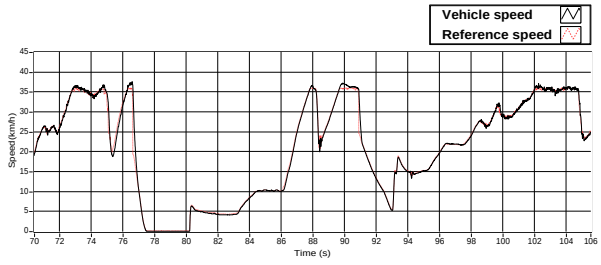


Fig. 10. Control performance of the speed controller

VIII. Discussion

This paper provides an overview of the autonomous vehicle Pharos, developed to participate in the 2010 Autonomous Vehicle Competition in South Korea.

From a broad perspective of view, Pharos's control software mirrors common methodology in the area of autonomous vehicle. However many of the individual modules have developed to achieve the high speed unmanned driving. Long range terrain perception and obstacle detection algorithm, real-time trajectory planning and obstacle avoidance algorithm, and high speed vehicle motion control method are developed and integrated all together. All the developed algorithms are extensively tested in the field to prove the reliability.

Even though our vehicle could successfully finish the race, there are many issues to improve our vehicle further. Only static environment was considered in perception. Pharos is unable to properly interact with moving objects. Perception range was limited within 30 m, which limited the maximum speed of our vehicle. In some case, to avoid an obstacle it was necessary to move up to 6 m to lateral direction from the reference path. If the perception range is limited, vehicle must reduce the approaching speed to avoid an obstacle which has large lateral offset from the reference path. We expect our methodology can allow us to perceive longer range than 30 m.

IX. Acknowledgments

The authors gratefully acknowledge the contribution of Pharos Racing Team for their passion to build our vehicle.

References

[1] M. Buehler, K. Iagnemma, S. Singh, *The 2005 DARPA grand challenge: the great robot race*, Springer Trats in Advnaced Robotics 36, Springer-verlag, Berlin; 2007.

[2] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, Vol 22, No. 6, 1989, pp. 46-57.

[3] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. "Practical Search Techniques in Path Planning for Autonomous Driving", *in Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics*, 2008.

[4] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance and Control*, Vol. 25, No. 1, 2002, pp. 116-129.

[5] A. Kelly, *A partial analysis of the high speed autonomous navigation problem*, Project report of "Perception for Outdoor Navigation" and "Unmanned Ground Vehicle System," Carnegie Mellon University, 1994.

[6] Y. Kuwata, G.A. Fiore, J. Teo, E. Frazzoli, and J.P. How, "Motion planning for urban driving using RRT", *in Proc. IROS*, 2008, pp. 1681-1686.

[7] S. M. LaValle and J. J. Kuffner. "Randomized kinodynamic planning". *International Journal of Robotics Research*, Vol. 20, No. 5, 2001, pp. 378-400.

[8] S. Thrun and et al., "Stanley: The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics*, Vol. 23, No. 9, 2006, pp. 661-692.

[9] C. Urmson and et al., "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, Vol. 23, No. 8, 2006, pp. 427-508.

[10] C. Urmson and et al., "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, Vol. 25, No. 8, 2008, pp. 425-466.

[11] http://www.hyundai-ngv.com/techcontest/, in Korean.

Generating State Lattice Motion Primitives for Differentially Constrained Motion Planning

Mihail Pivtoraiko and Alonzo Kelly

***Abstract*— We propose a method of generating motion primitives for motion planning under differential constraints. These primitives lend themselves particularly well to off-line precomputation. In this manner, they are attractive due to their capacity to capture the mobility constraints of the robot and to establish a state sampling policy that is conducive to efficient search. The first objective allows encoding mobility constraints into primitives, thereby enabling fast unconstrained search to produce feasible solutions. The second objective enables high quality (lattice) sampling of state space, further speeding up exploration during search. A number of techniques, novel in differentially constrained planning, are enabled with this approach, including incremental search, efficient bi-directional search and incremental sampling.**

I. INTRODUCTION

Differentially constrained motion planning is an open and challenging area of research in part due to the complexity involved in representing the feasible motions of the system. Since it is intractable to search the continuum of a system's reachability, approximation methods based on sampling must be developed. To that end, a number of approaches to sampling system control space have been developed over the last several decades [1–7]. Related to that, there has been significant interest recently in developing *motion primitives*, sequences of control samples that are frequently designed *a priori*, in other words *pre-computed*, and represent *feasible* motions, i.e. those that satisfy the constraints of the system. Motion planners can utilize these primitives to enact efficient search in state space by ignoring system constraints, instead focusing on the environment and other constraints – thereby improving efficiency of the planning. The role of primitives in planning and the importance of their quality have been motivated both in deterministic [6, 8, 9] and randomized [10] planning domains. Their importance was also noted in the related area of reactive obstacle avoidance in the context of mobile robot navigation [5, 11, 12]. A number of popular approaches to kinematic and kinodynamic planning can readily incorporate primitives in their design. The requisite *local planner* in [13–15] can be implemented as a process that chooses an appropriate element from a set of primitives [10]. In deterministic approaches [1, 8, 9, 16–18], the vertex expansion (set of edges emanating from a vertex) can be a pre-determined set of primitives based on the state value that the vertex represents.

This work was supported in part by the NASA Graduate Student Researchers Program Fellowship.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {mihail, alonzo}@cs.cmu.edu

Motion primitives have been designed in the past through sampling control space in such a way as to result in good sampling in state space in terms of discrepancy, dispersion or path diversity [5, 6, 8, 11, 12, 19]. We refer to this line of work as *control-sampling* primitives. In general, designing such primitives is difficult due to the complexity of the relationship between the robot's control and state spaces under kinematics and dynamics constraints. On the other hand, a reverse design process has been proposed [20, 21]. First, an attractive sampling rule in state space, perhaps one that is convenient and efficient for the planning problem (e.g. commensurate with the robot's world model, such as an occupancy grid), is established. Then, we compute the controls that steer the system between these samples using a boundary value problem (BVP) solver. The approach can be viewed as a way of extending the Lazy LRM [22] to handle kinematics and dynamics constraints by leveraging the related research in BVP. Such solvers are available for a variety of systems, such as car-like [23], chained form [19, 24], as well as in rough terrain [25] and dynamics [26, 27] settings. A simple example of a set of car-like primitives in a three-dimensional (2D position and heading) state space is shown in Figure 1. A functional planner would require such a set for each of the 8 discrete values of heading, the multiples of $\pi/4$.

The benefits of this type of primitives are four-fold. First, by providing the freedom to choose an arbitrary sampling of state space for primitive endpoints, quality state sampling policies (low discrepancy and dispersion) may be utilized, leading to efficient exploration of state space during search. As Figure 1 illustrates, the resulting primitives may feature good path diversity as well. Second, under certain assumptions, such as flat and uniform terrain for the example above, this freedom allows designing primitives to be position-invariant. Experience with fielded applications demonstrated that position-invariance assumptions are often satisfied in

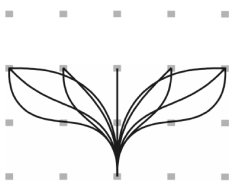


Fig. 1. The state lattice motion primitives are regularly arranged in state space. To design them, a convenient state sampling rule is chosen (e.g. a low discrepancy lattice), then a BVP solver is used to connect the samples via feasible motions.

practice, as long as a trajectory following controller absorbs external perturbations [28]; if necessary, trajectory post-processing may also be performed [29]. Once computed, position-invariant primitives can be utilized anywhere in search space, thereby moving integration of the controls to planner design phase. This is more efficient than affine invariance [8], since primitive transformation during search is limited to translation. Third, special reachability tree pruning rules can be easily designed. In contrast to control-sampling primitives, where primitive endpoints are dense in state space, state lattice ones yield a structure, where all paths leading to a region in state space also lead to a unique state value, as illustrated in Figure 2. This structure can be exploited to attain unprecedented search efficiency in the area of kinodynamic planning, including incremental search (a potential to speed up planning by orders of magnitude) [30], incremental sampling [22] and bi-directional search [31]. Finally, the freedom in state sampling may allow fitting the search space to the known structure of the environment. This strength of the approach was utilized recently to fit search spaces to such settings as parking lots [9], roads [32], mines [17] and indoor environments [33].

One drawback that may be experienced with the proposed primitives is the potentially significant computation that may be required to design this type of primitives (perhaps running the BVP solver repeatedly). However, this computation is off-line and does not affect the runtime of the planner. As another potential difficulty in certain applications, the constraint that the motions are arranged in a particular manner may conflict with other relevant objectives. For example, minimizing the length of primitives may be helpful for planning amidst dense obstacles, since shorter motions are less likely to be obstructed [6, 34]. Meeting such an objective may be more challenging if a constraint on endpoint arrangements is placed. Finally, even though the motions computed using lattice primitives are feasible and may be executed by the system *verbatim*, most physical systems suffer from inaccuracy in control leading to trajectory following error. Some applications may still require a trajectory-following controller, motivated above to satisfy position-invariance assumptions in rough terrain and similar scenarios. Significant disturbances in the environment, such as slopes or wind may be accounted for as additional state variables. The recommendation for a trajectory following controller does not offset the value of planning feasible motions, since non-feasible ones are more difficult or impossible to follow.

This paper provides a more general exposition of lattice primitives introduced in [20] and motivates them beyond field robotics [28]. It also proposes new applications of these primitives in incremental and randomized search (Section II), as well as *D* decomposition*, a novel algorithm to apply elements of D* search [30] to representation design, whereby near-minimal sets of lattice primitives are generated automatically (Section III). Experimental validation is discussed in Section IV.

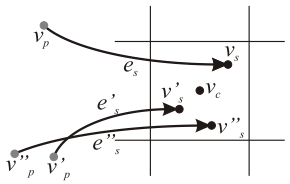


Fig. 2. State cell predecessors. Three control-sampling primitives, edges $\{e_s, e'_s, e''_s\}$, emanate from their corresponding predecessor vertices $\{v_p, v'_p, v''_p\}$ and arrive at successor vertices $\{v_s, v'_s, v''_s\}$.

II. Motion Planning with Lattice Primitives

In this section, we discuss the specifics of applying the state lattice primitives in planning using two prominent classes of search algorithms: deterministic (e.g. A*, D* [30] and their variants) and randomized (e.g. PRM [14], EST [13], RRT [15] and their variants). The planning problem is specified with a seven-tuple $(X, X_{free}, x_{init}, x_{goal}, U, f, c)$. The robot *state space*, $X \subset \mathbb{R}^n$, is an n-dimensional compact differentiable manifold, equipped with a metric ρ. $X_{free} \subseteq X$ is the set of states that satisfy global constraints (e.g. control bounds, obstacle avoidance, etc.). The boundary conditions for the planning problem are $x_{init} \in X_{free}$ and $x_{goal} \in X_{free}$. The set of robot controls U contains the inputs that the system accepts. The function f is the system model (equation of motion) and encodes kinematics and dynamics constraints: $\dot{x} = f(x,u)$, where $x \in X, u \in U$. The function $c : U \times X \to \mathbb{R}$ specifies the cost of executing a control $u \in U$ in X. The solution to the planning problem is a control $u_s : [t_0, t_f] \to U$, where t_0 is the starting time and t_f is the final time, such that $c(u_s, x_{init})$ is minimized. The corresponding path $\pi_s : [t_0, t_f] \to X_{free}$ (obtained by integrating $f(x_{init}, u_s)$) satisfies $\pi_s(t_0) = x_{init}$ and $\pi_s(t_f) = x_{goal}$.

Finding the exact solution involves optimization over the continuum of X and U, a difficult problem because of obstacles and local optima in X. Instead, it is common to establish pruning rules that reduce the system's reachability in X and U to discretized representations, often structured as graphs. We assume a directed graph $G = V \cup E$, where V is a set of vertices, representing samples in X, and E is a set of edges, representing samples in U. Each edge is one of the pre-computed, feasible primitives. The dimensionality of V is chosen so that a concatenation of edges is also a feasible motion. The least-cost path in the graph is the solution to the planning problem.

A. Deterministic Search

The strengths of deterministic, exhaustive search include attractive guarantees, such as optimality (under certain conditions, such as heuristic admissibility) and resolution-completeness. One drawback, however, is the so-called "curse of dimensionality", the exponential growth of complexity with dimension of the search space. Nevertheless, this search technique remains attractive for systems that can be modeled well in a few dimensions, including car-like [9], tracked [17], flying [8] and other systems of practical interest.

Such approaches to deterministic nonholonomic planning typically guarantee feasibility by elaborating the primitives $e_s \in E$ by choosing a control $u_s \in U$ and integrating $f(v_p, u_s)$ for a certain Δt. The successor vertex v_s is established at the endpoint of e_s. Since, in general, such edges and their endpoints will be dense in X, such planners attempt to prune the edges that are very similar, redundant or otherwise do not contribute to efficient exploration of X. For example, if the endpoint of a certain edge e_s is a vertex v_s, then a second edge e'_s terminating in v'_s is discouraged if distance $\rho(v_s, v'_s)$ is small. To this end, these approaches establish a discretization of X into cells, as shown in Figure 2. If a cell contains v_s, then e'_s is ignored if v'_s would occupy the same cell. The proposed lattice primitives may be used identically, except that the need to detect similar edge endpoints is eliminated, since all motions are designed to arrive at specific state values, e.g. v_c in Figure 2. A similar pruning rule is in effect in this setting, except that it is developed offline during search space design.

Well-informed search heuristics have the potential to increase the efficiency of the search substantially. Developing good heuristics for planning with differential constraints is a challenging problem, and various approaches are being developed [17, 35]. By position invariance of lattice primitives, applications are enabled to pre-compute the free-space costs of motions, stored in a look-up table, leading to the perfect heuristic in terms of mobility constraints [36].

1) Incremental Search: In many applications featuring physical robots, it is beneficial to perform incremental search: once a plan is computed, it is efficiently modified (by reusing previous computation) should new information about the environment invalidate it [30]. This enables the planner to react quickly to frequent changes of the world model, including those due to uncertainty and noise of the perception, localization and other systems. This type of search is a standard component in many fielded robotics systems, since plan repair can be vastly more efficient than replanning from scratch.

The reachability pruning schemes above, necessary for control-sampling primitives, are not fully compatible with incremental search. The operation of such search requires an ability to enumerate the edges that lead to a particular state, e.g. the edges $\{e_s, e'_s, e''_s\}$ leading to a cell containing v_s in Figure 2. A key component of incremental search algorithms is the *rhs*-value, the one-step lookahead cost value, that is defined as [30]:

$$rhs(v_s) = \begin{cases} 0 & \text{if } v_s = x_{init} \\ \min_{v_p \in Pred(v_s)} (g(v_p) + c(v_p, v_s)) & \text{otherwise} \end{cases} \quad (1)$$

where $g(v_p)$ denotes the cost of the vertex v_p, $Pred(v_s)$ is the set of the predecessors of v_s, e.g. $\{v_p, v'_p, v''_p\}$ in Figure 2. One of the ways of reusing previous computation is the capacity to pick a different predecessor of a state in the event that the current predecessor edge increases its cost. However, as depicted in Figure 2, the predecessor edges cannot be swapped in general because the distance between their endpoints $\rho(v_s, v'_s) \neq 0$ (by their density). This issue is resolved with lattice primitives: this distance is zero, since all successors converge to the identical v_c. Note that this is similar to traditional applications of incremental search in grids.

Moreover, in grids, it is typically easy to determine the set of cells (and, consequently, edges) that are affected when a region in workspace changes cost. With arbitrary motion primitives, this computation is more involved, since the edges may span several cells. As discussed in [28], this computation can be done *a priori* by virtue of the lattice structure. Once we compute the *swaths* of all primitives, we collect and store a list of edges that pass through the origin cell. By position invariance, this list may be reused anywhere in the search space.

2) Incremental Sampling: The advent of randomized planning in recent years has spurred inquiry into effective sampling methods that avoid the "curse of dimensionality" while offering better solution quality and completeness guarantees than standard randomized approaches. Deterministic incremental sampling techniques have been proposed as viable alternatives [22]. One of them is the Halton points, a d-dimensional generalization of the van der Corput sequences of d bases, one for each coordinate [37]. A basic version of such incremental sampling in square grids can be thought of as increasing discretization resolution by a factor of 2^{di}, $i \in \mathbb{N}$.

The planners that do not enforce structure in edge connectivity would be required to regenerate the plan from scratch every time the sampling resolution is incremented. However, lattice primitives enable the reuse of previous computation via the same mechanism that allows incremental search. Due to regular structure, the connections between primitives belonging to different resolution levels become trivial, thereby allowing the results of planning at different resolution to be reused. One application of this approach is *anytime* planning, where the quality of the computed plan is improved with more computation.

B. Randomized Search

Lattice primitives may be utilized in randomized search in a manner that is similar to other types of primitives [10]. As suggested in Section I, the local planner component in [13–15] and similar planners may be designed to choose an element of a set of primitives that is a good fit to extend the tree or the graph towards a random sample. However, by virtue of the regular structure of the state samples, lattice primitives would enable additional capacity to execute parallelized kinodynamic planning, e.g. bi-directional [31], as well as a series of independent searches [38]. Figure 3 illustrates that, unlike control-sampling primitives that would likely require multiple BVP solutions to connect partial planning results, the layout of lattice primitives makes this connection automatic.

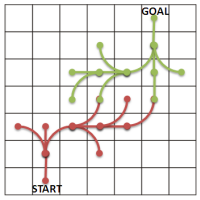


Fig. 3. Bidirectional search with lattice primitives. The BVP problem of connecting the leaves of the two trees is eliminated with lattice primitives that feature regularity of endpoints in state space.

III. Designing Lattice Primitives

Having motivated the proposed type of primitives, we suggest a principled approach to designing them given the planning problem formulated in Section II. Assuming a system model and a corresponding BVP solver, we discuss this design in three stages: making a choice of the dimensions to include in the state space representation, developing the sampling rule in that state space, and designing a compatible, near-minimal set of lattice primitives that is a good representation of the system's reachability. The first two stages (Section III-A) determine the set of controls $U_l \subset U$ that can possibly be represented with such primitives. Generally, this set is infinite; Section III-B is dedicated to developing a near-minimal primitive set $E_a \subset U_l$ that, when used as the vertex expansion in search, will reconstruct a good approximation to U_l. More precisely, a planner, based on optimal (exhaustive) search and equipped with E_a, will be able to compute the motions in U_l (preserve completeness) and will guarantee bounds on suboptimality of these motions w.r.t. U_l. An algorithm that computes E_a automatically is presented.

A. State Space Sampling

In general, the problem of selecting the minimal number of dimensions that adequately represent the planning problem is quite challenging. In the case of designing lattice primitives, this issue is influenced by the choice of the BVP solver. In case the solver does not fix the dimensionality, an iterative dimensionality reduction process may be undertaken. Once a set of lattice primitives is designed at the highest dimensionality, it may be repeated with one of the dimensions removed. The process iterates until the loss of representation quality exceeds application tolerances.

Once state dimensionality is fixed, we develop a state sampling rule using two principles. First, it is beneficial for the sampling rule to minimize discrepancy or dispersion [22]. Grids and similar regular lattice structures typically minimize these measures, and they are frequently used in this setting. Second, among similarly performing search spaces, those with more coarse sampling are preferred. This is an Occam's razor statement: a simpler approach is likely to lead to a solution that is easier to develop and test. Since controls are induced by state sampling in this setting, it is beneficial to choose state samples that reduce the cost of controls, e.g. lead to a greater number of straight-line motions. The above principles are purposefully broad: each application imposes specific requirements on state sampling. For example, sampling of position and orientation variables of robots is often closely related to other design specifications, such as the fidelity of perception information and control accuracy of the vehicle.

B. Primitive Set Decomposition

Even though the representable set of controls U_l is infinite, the reachability of many systems of practical interest can be captured well by analyzing a finite, albeit very large, subset $\hat{U}_l$. For example, for car-like robots, we could define $\hat{U}_l$ as the set of motions that are contained in a region (centered around the robot) that is much larger in extent than the robot's minimum turning radius. This motion set will include many maneuvers that the robot is capable of executing, including multi-point turns in close quarters. Next we develop an explicit and exact representation of $\hat{U}_l$ as a graph $\hat{G}_l = \hat{V}_l \cup \hat{E}_l$, as justified in Section II.

By the given lattice state sampling rule, the set $\hat{V}_l$ is known. Theorem 1 develops a primitive set E_O that generates $\hat{E}_l$, a superset of $\hat{U}_l$, when used as the Dijkstra's vertex expansion; free space is assumed below, unless otherwise noted. More precisely, E_O is a set of primitive sets defined for all possible trajetory initial states in $\hat{V}_l$, up to the invariant dimensions (e.g. position). For example, different values of heading in Figure 1 would require different vertex expansions; E_O can be viewed as the union of the corresponding primitive sets.

Theorem 1: Suppose origin vertices $O \subset \hat{V}_l$ are chosen (up to invariant dimensions). For every vertex $v_i \in \hat{V}_l$, an edge from each element of O to v_i is computed using the BVP solver and added to E_O (initially empty). When used as the Dijkstra's vertex expansion, E_O will search (equivalently, generate) a $\hat{G}_l$ such that $\hat{E}_l \supseteq \hat{U}_l$.

Proof: First, by construction, we conclude that E_O contains $\hat{V}_l$ as endpoints of its primitives. Using E_O as the Dijkstra's vertex expansion amounts to replicating its edges at every $v_i \in \hat{V}_l$. If, per connectivity of $\hat{U}_l$, a certain v_i connects to a set of vertices $\{v_j\}$, then E_O, when replicated at v_i, will connect it to at least the same vertices, since $\{v_j\} \subseteq \hat{V}_l$. Thus, the process of replicating E_O at $v_i \in \hat{V}_l$ generates at least the edges present in $\hat{U}_l$, and therefore the induced set of edges $\hat{E}_l \supseteq \hat{U}_l$. ■

Using E_O as the vertex expansion represents an extreme of quality-complexity trade-off. The cost of the capacity to explore at least all of $\hat{U}_l$ during search is a very large branching factor, $|E_O|$. Next we discuss an approach to manage this trade-off by computing an approximation primitive set $E_a \subset E_O$ of much smaller cardinality, while guaranteeing bounded suboptimality of computable motions w.r.t. $\hat{U}_l$ in terms of arbitrary notion of cost.

This process attempts to decompose each motion in $\hat{E}_l$ into two or more other motions that are also in $\hat{E}_l$. Decomposition (Figure 4) is allowed only if the concatenation of the components is within a user-specified threshold on cost increase, defined as a cost ratio $c_t > 1$ of the concatenated motion vs. the original one:

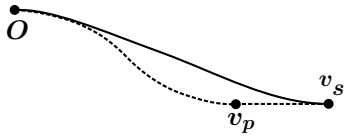


Fig. 4. Motion decomposition. A motion (solid line) is approximated by a concatenation of two shorter motions (dashed lines). The ratio of their combined cost to the original cost is constrained not to exceed a user-specified threshold.

$$g(v_p) + c(v_p, v_s) \leq c_t g(v_s) \tag{2}$$

The component motions that can be reused to generate other motions are collected into E_a. E_a is designed to be capable of generating every one of the motions in $\hat{E}_l$, within the specified cost increase threshold.

A brute-force approach to decomposing $\hat{E}_l$ by enumerating all possible motion decompositions would be exponential in $\left|\hat{E}_l\right|$ and therefore prohibitively expensive. We propose two greedy algorithms that produce near-minimal E_a, given $\hat{E}_l$ and c_t.

1) Leave-one-out Decomposition: This algorithm decomposes the motions in $\hat{E}_l$ in decreasing order of their cost. This implements a heuristic on managing the dependencies of component motions, namely that the higher-cost motions are likely to be decomposed by lower-cost motions. Each motion e_s to be decomposed is selected and removed from $\hat{E}_l$. Next, optimal A* search is used to compute a motion from O to the endpoint of e_s using vertex expansion $\hat{E}_l \setminus \{e_s\}$. If the least cost motion represents a cost increase greater than c_t, the e_s is reinserted into $\hat{E}_l$, since it cannot be decomposed satisfactorily. Otherwise, it is left out of $\hat{E}_l$, and the algorithm repeats by selecting the next motion to decompose. Very large values of c_t will lead to a degeneracy where almost all motions are decomposed. This condition can be detected by observing the resulting E_a.

2) D Decomposition:* The previous algorithm has a disadvantage in complexity: it requires running A* "from scratch" $\left|\hat{E}_l\right|$ times, in graphs with large (albeit reducing) branching factors. Inspired by the capacity of incremental search algorithms, e.g. D* Lite [30], to prevent repetitive "from scratch" search by virtue of reusing computation, we propose an alternative with much better runtime (Algorithm 1). It does a single Dijkstra's elaboration of E_O vertex expansion to explore the extent of $\hat{U}_l$ (line 2) and then performs the decomposition analysis from Section III-B.1 while reusing the collected vertex predecessors, similar to evaluating the *rhs*-value (1). In this manner, unlike the traditional application in planning, the principles of the D* algorithm are used here for search space design.

Once the predecessors are computed, only those that can possibly yield c_t-decompositions are retained (lines 3-7). If a state has only a single predecessor, it implies by construction that the one and only motion that connects it to the origin is the original edge in $\hat{E}_l$, and motion decomposition cannot succeed. This edge is entered into E_a. Since the motions

Input: finite control set $\hat{E}_l$, cost threshold c_t
Output: approximating control set E_a

1 $E_a = \emptyset$;
2 Run Dijkstra's search with $\hat{E}_l$, starting at O;
3 **foreach** $e_s \in \hat{E}_l$ **do**
4 $\quad Pred_{c_t}(v_s) = \{v_p, \text{ s.t. } g(v_p) + c(v_p, v_s) \leq c_t g(v_s)\}$;
5 $\quad db(v_s) = c_t$;
6 $\quad$ **if** $|Pred_{c_t}(v_s)| = 1$ **then**
7 $\quad\quad E_a = E_a \cup e_s$;
8 $E_{sort} = \text{sort}(\hat{E}_l)$;
9 **foreach** $e_s \in E_{sort}$ **do**
10 $\quad$ **if** $|Pred_{c_t}(v_s)| = 1$ **then**
11 $\quad\quad E_a = E_a \cup e_s$;
12 $\quad\quad$ continue;
13 $\quad$ **foreach** $v_p \in sort(Pred_{c_t}(v_s))$ **do**
14 $\quad\quad$ **if** $e_{v_p,v_s} \in E_a$ **then**
15 $\quad\quad\quad$ adjust_threshold(v_p, v_s);
16 $\quad\quad\quad$ goto 9;
$\quad\quad v_p^* = \underset{v_p \in Pred_{c_t}(v_s)}{\text{argmin}} \; g(v_p) + c(v_p, v_s)$;
17
18 $\quad$ adjust_threshold(v_p^*, v_s);
19 $\quad E_a = E_a \cup e_{v_p^*, v_s}$;

Algorithm 1: D* Decomposition.

Input: vertices v_p and v_s
Output: $db(v_p)$, $Pred_{c_t}(v_p)$, and E_{sort} modified

1 $\alpha = (g(v_p) + c(v_p, v_s))/g(v_s)$;
2 $c_t' = \frac{db(v_s)g(v_s) - c(v_p, v_s)}{\alpha g(v_s) - c(v_p, v_s)}$;
3 **if** $c_t' < db(v_p)$ **then**
4 $\quad db(v_p) = c_t'$;
5 $\quad Pred_{c_t}(v_p) = \{v_p', \text{ s.t. } g(v_p') + c(v_p', v_p) \leq db(v_p) g(v_p)\}$;
6 $\quad$ **if** $g(v_p) \geq g(v_s)$ **then**
7 $\quad\quad E_{sort} = E_{sort} \cup e_p$;

Algorithm 2: adjust_threshold function.

that form decompositions may be further decomposed themselves, we keep track of per-vertex cost thresholds with a database db; thresholds for all vertices are initially set to c_t on line 5.

Line 8 follows the cost-sorting heuristic discussed in Section III-B.1. The next line selects edges in decreasing order of cost of their destination endpoints. Lines 10-12 re-evaluate the number of admissible predecessors. Line 13 enumerates the options for decomposing the edge e_s leading to v_s. The algorithm attempts to select the decomposition that is likely to minimize the final E_a. To this end, it uses two other heuristics. The first one is sorting potential decompositions in increasing order of cost (sort on line 13), in an attempt to choose a decomposition with cumulative cost that is as close as possible to the original motion. The second heuristic is preferring a decomposition, in which one

of the segments is already found in E_a (lines 14-16), in an attempt to avoid adding new elements to E_a.

Once an edge is decomposed into two segments connecting at vertex v_p, the segment leading to v_p may in turn be decomposed; however the cost threshold for its decomposition may have to be different from c_t. To see this reasonsing, suppose path length is taken as cost (Figure 4). Using the cost sorting heuristic, the longer, original motion (terminating in v_s) will be decomposed before the first segment of decomposition (terminating in v_p). When the latter is in turn selected for decomposition, we rewrite (2) as:

$$c_t' g(v_p) + c(v_p, v_s) \leq c_t g(v_s) \tag{3}$$

Depending on the value of $c(v_p, v_s)$, the required cost increase threshold c_t' for the recursive decomposition of v_p may be $c_t' < c_t$. Thus, as soon as a decomposition relationship is established between the vertices, the necessary reduction of cost threshold of the predecessor vertex is computed with the function *adjust_threshold*. If the algorithm gets to line 17, no decomposition opportunities have been selected by the heuristics. This line selects a decomposition with the least-cost predecessor; it is added to E_a on line 19.

Algorithm 2 presents the *adjust_threshold* function. In line 2, it computes c_t', the new cost threshold for the predecessor vertex v_p. If this value is less than the previous one, it is recorded on line 4, and the list of admissible predecessors is reviewed on line 5. Lines 6-7 address the case where the cost sorting heuristic is incorrect, and the predecessor v_p has equal or greater cost than v_s. In this case, any changes done to v_p need to be propagated to its potential predecessors, so the next iteration of Algorithm 1 must select v_p's edge, e_p (line 3).

IV. Experimental Results

We present experimental validation results of kinodynamic planning using two examples: a double integrator system inspired by [16] and a car-like system with complex dynamics. We also describe a multi-query BVP approach (Section IV-B) that is helpful for implementing the second example, presented in Section IV-C. Both examples were developed by applying the design principles in Section III (Figure 6). The primitives, developed with Algorithm 1, are then used to perform incremental search by utilizing the unmodified D* Lite algorithm [30] (Figure 7).

A. Double Integrator

The *double integrator* is a simple dynamics system, where the acceleration is controlled directly. Formally, it is governed by a second-order differential equation $\ddot{x} = u$, where $x, u \in \mathbb{R}$. We look at a one-dimensional example here, although more dimensions, similarly defined, can be added easily due to their independence. The state transition equation $\dot{x} = f(x,u)$ can be written as $(\dot{x}_1, \dot{x}_2) = (x_2, u)$. Thus, the terminal states of a trajectory, x_{init} and x_{goal} are specified by the corresponding positions and velocities. A discrete-time model with controls given by $u = \{-1, 0, 1\}$ has reachability tree trapped on a lattice (Figure 5a).

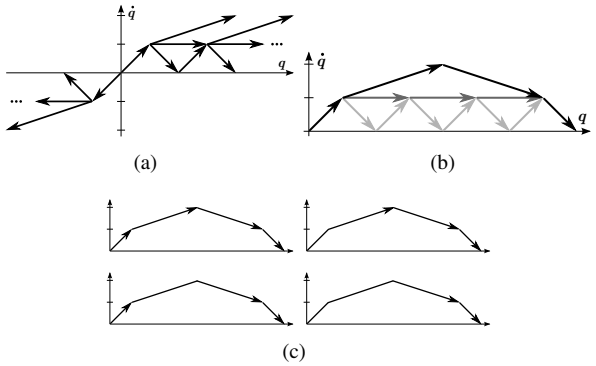


Fig. 5. Double integrator system example.

To illustrate the application of Algorithm 1 to this system, consider an example in Figure 5b. The system starts at zero position and zero velocity and attempts to reach a specified position value with zero terminal velocity. The trajectory in this example is the sequence of black arrows in part b) of the figure. Part c) shows a few of the elements of the set $\hat{E}_l$ for this example; the whole set is prohibitively large to illustrate. Given this input, the algorithm does Dijkstra's (line 2), which generates the sets of admissible predecessors of the vertices. This set for the goal state, $Pred_{c_t}(x_{goal})$ is shown in gray in Figure 5b. Since the edges between all the states in this set come from the original selection of edges allowed in this problem setup, we observe that the solution E_a contains the same set of edges we started with (Figure 5a). In the case of this simple problem, the algorithm is able to find the optimal solution (that is, resulting c_t ratio is 1.0), however it is usually not the case for more realistic problems, as suggested by the following sections. The solution path (black line in part b) of the figure) can be constructed with the replanning search algorithms we consider here.

B. Multi-Query BVP Approach via Reachability Analysis

Since the BVP solver was not available for this system, we utilized an alternative approach; it is described here because it illuminates the experimental setup. First, a dense reachability tree of the system was generated via significant computation using high-resolution, regular sampling in two-dimensional control space, consisting of angular wheel velocity and the steering angle. A reachability pruning technique [1, 8, 18] was utilized via high-resolution, regular sampling in seven-dimensional state space, consisting of 2D position, heading, steering angle, longitudinal, lateral and angular velocities of robot body. The generated reachability was analyzed to engineer appropriate dimensionality of representation. It was observed that the lateral and angular velocity dimensions could be dropped without a significant loss of representation quality: the subset of reachability with these variables considered to be zero (per pruning resolution) was henceforth considered. This reasoning was motivated by dimensionality reduction considerations in Section III-A.

A sampling scheme of the remaining five dimensions was developed via an application-minded approach. First, the chosen state sampling resolution was lower than the reachability resolution by an integer multiple. Since minimum turning radius was small compared to vehicle size, the position was sampled as a grid with cell size equal to approximately half of min. turning radius. Heading was sampled in 16 non-uniform values: half were multiples of $\pi/4$, and the other half were related to $\arctan(1/2)$ to maximize straight paths toward nearby cells. Only extremal values of steering angle (including 0) and longitudinal velocity (three and two, respectively), were chosen in order to explore the envelope of the system dynamics. This sampling rule generated another, yet smaller, subset of system reachability, denoted U_l in Section III-B. Since the state resolution was a multiple of the reachability pruning resolution, the endpoints of the primitives were close to their respective state cell centers. Those that were not sufficiently close were improved via gradient descent optimization by treating the durations of control space samples, comprising the trajectory, as variables and the distance of the end-point to cell center as the objective. This was a significant computation, since gradient estimation involved repeated execution of the physics simulation.

C. Car-like Robot with Dynamics

The system in this example is a wheeled robot with three driven wheels, one of which steers, as shown in Figure 7b. The system is simulated using the Open Dynamics Engine^{TM}software; the system model is not available in closed form. The vehicle has significant mass, is capable of achieving high speeds and is placed on a very slippery flat surface to highlight the effects of dynamics, such as significant drift, sliding sideways, etc.

A comparative study of lattice and high-diversity primitives [5, 6], based on A* search, showed that the difference in performance in terms of runtime, solution quality and completeness of both types of primitives is less than statistically significant.

The benefit of the principled approach here is that the length of primitives is selected automatically, while it is fixed for path diversity primitives.

Next we describe experimental validation of incremental search in this context, although other planning approaches may benefit from such primitives, as discussed in Section II. Figure 7 illustrates an example where a new obstacle invalidated a segment of the previously computed trajectory. D* modified the plan efficiently by limiting vertex expansions to a small neighborhood. The initial plan was computed in 1.42 seconds (on commodity hardware), and was repaired in 0.35 seconds – a nearly 4-fold speedup with respect to re-planning from scratch.

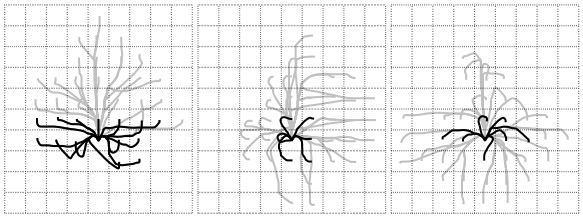


Fig. 6. Automatic pruning of primitive sets. Several subsets of primitives, generated for the given system, are shown; abundance of high curvature is due to extreme dynamics. Top and bottom rows include primitives at low and high velocity, resp. The columns show the primitives with final headings 0°, 90° and 180° (left to right). Gray motions have been pruned, as concatenations of black motions can replicate them within specified cost increase threshold.

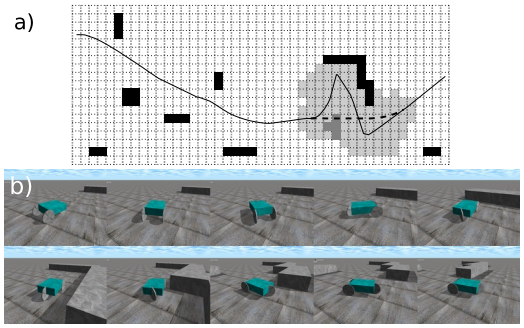


Fig. 7. Kinodynamic incremental planning. Robot is avoiding a number of obstacles (black cells), while traveling at high speed on slippery surface. A new obstacle (dark gray cells) is discovered and invalidates a segment of the previous trajectory (dotted line). D* repairs the path by expanding states (light gray) only in the affected region.

V. Conclusions and Future Work

We discussed a type of primitives that is designed via regular sampling in state spaces. These primitives are precomputed to meet two objectives: to capture the mobility constraints of the robot as well as possible and to establish a state sampling policy that is conducive to efficient search. The first objective allows encoding mobility constraints into primitives, thereby enabling fast unconstrained search to produce feasible solutions. The second objective enables high quality (lattice) sampling of state space, further speeding up exploration during search. We further discuss several novel results enabled by using such primitives for kinodynamic planning, including incremental, bi-directional search and incremental sampling. Future work includes identifying new state and control sampling techniques that further improve properties of planning in deterministic and randomized domains.

VI. Acknowledgments

The authors gratefully acknowledge the contribution of the NASA Graduate Student Researchers Program.

References

[1] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, no. 2-4, pp. 121–155, 10 1993.

[2] A. Stentz and M. Hebert, "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol. 2, no. 2, pp. 127–145, 1995.

[3] B. Donald and P. Xavier, "Near-optimal kinodynamic planning for robots with coupled dynamics bounds," in *Proc. Symp. IEEE Int Intelligent Control*, 1989, pp. 354–359.

[4] C. Green and A. Kelly, "Toward optimal sampling in the space of paths," in *Proc. of the Int. Symp. of Robotics Research*, 2007.

[5] M. S. Branicky, R. A. Knepper, and J. J. Kuffner, "Path and trajectory diversity: Theory and algorithms," in *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 2008, pp. 1359–1364.

[6] L. H. Erickson and S. M. LaValle, "Survivability: Measuring and ensuring path diversity," in *Proc. IEEE International Conference on Robotics and Automation ICRA '09*, 2009, pp. 2068–2073.

[7] R. A. Knepper, "On the fundamental relationships among path planning alternatives," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, June 2011.

[8] J. Go, T. D. Vu, and J. J. Kuffner, "Autonomous behaviors for interactive vehicle animations," *Graph. Models*, vol. 68, no. 2, pp. 90–112, 2006.

[9] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 8 2009.

[10] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control and Dynamics*, vol. 25, no. 1, 2002.

[11] C. Green and A. Kelly, "Toward optimal sampling in the space of paths," in *13th International Symposium of Robotics Research*, 2007.

[12] M. Lau and J. J. Kuffner, "Behavior planning for character animation," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005, pp. 271–280.

[13] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. Conf. IEEE Int Robotics and Automation*, vol. 3, 1997, pp. 2719–2726.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[16] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[17] X. Fan, S. Singh, F. Oppolzer, E. Nettleton, R. Hennessy, A. Lowe, and H. Durrant-Whyte, "Integrated planning and control of large tracked vehicles in open terrain," in *Proceedings of the International Conference on Robotics and Automation*, 2010.

[18] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schader, M. Thuy, M. Goebl, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team annieway's autonomous system for the darpa urban challenge 2007," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615 – 639, 8 2008.

[19] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi, "Motion planning through symbols and lattices," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '04*, vol. 4, 2004, pp. 3914–3919.

[20] M. Pivtoraiko and A. Kelly, "Generating near-minimal spanning control sets for constrained motion planning in discrete state spaces," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, August 2005, pp. 3231–3237.

[21] ——, "Kinodynamic motion planning with state lattice motion primitives," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 2172–2179.

[22] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[23] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[24] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, May 1993.

[25] T. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[26] J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun, "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving," in *Proceedings of the International Conference on Robotics and Automation*, 2010.

[27] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Proceedings of the Neural Information Processing Systems Conference*, 2007.

[28] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[29] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning," *Robot motion planning and control*, 1998.

[30] S. Koenig and M. Likhachev, "D*-Lite," in *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 476–483.

[31] J. Kuffner, J. J. and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, vol. 2, 2000, pp. 995–1001.

[32] M. Rufli and R. Siegwart, "On the design of deformable input- / state-lattice graphs," in *Proceedings of the International Conference on Robotics and Automation*, 2010.

[33] M. Rufli, D. Ferguson, and R. Siegwart, "Smooth path planning in constrained environments," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '09*, 2009, pp. 3780–3785.

[34] P. J. Leven, "A framework for real-time path planning in changing environments," Ph.D. dissertation, University of Illinois, Champaign, IL, USA, 2001, adviser-Hutchinson, Seth.

[35] T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[36] M. Pivtoraiko and A. Kelly, "Constrained motion planning in discrete state spaces," in *Field and Service Robotics*, 2005, pp. 269–280.

[37] J. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, no. 1, pp. 84–90, December 1960.

[38] M. Likhachev and A. Stentz, "R* search," in *AAAI*, 2008.

Detection of Moving and Stationary Objects at High Velocities using Cost-Efficient Sensors, Curve-Fitting and Neural Networks

Florian Mirus[1], Jürgen Pfadt[1], Christian Connette[1], Björn Ewert[2], Dietmar Grüdl[2], Alexander Verl[1]

***Abstract*— In recent years, driver-assistance systems have emerged as one major possibility to increase comfort and – even more important – safety in road traffic. Still, cost is one major hindrance to the widespread use of safety systems such as lane change or blind spot warning. To facilitate the widespread adoption of such assistance systems, thus increasing safety for all traffic participants, the use of cost-efficient components is of crucial importance.**

This paper investigates the usage of cost-efficient, widely used ultrasonic sensors for blind spot warning at high velocities. After discussing the requirements and setup of such a system a model-based approach for the detection of moving and stationary objects is outlined. The sensor-signal is compared with a precalculated curve data base and the correlation-coefficients are feeded into a neural network. To revise its performance the concept at hand is qualitatively and quantitatively evaluated in real road traffic situations under different driving conditions.

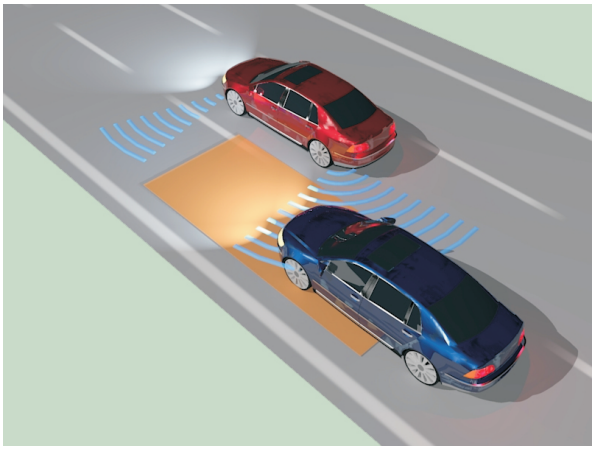


Fig. 1. Approaching car in the host vehicle's blind spot

I. Introduction

During the last decade, autonomous driving has made a huge jump from the first DARPA challenge [1] over the last Urban Challenge [2], [3], [4] to the latest experiments of Google in the field of autonomous driving. Although legal considerations and costs might be an insurmountable obstacle for a long time, driver-assistance systems have emerged as one major possibility to increase comfort and safety in road traffic [5].

Besides mainly introspective systems such as ABS and ESP recent developments [6] build more and more on exteroreceptive sensors to detect and react on potentially dangerous situations. To facilitate the widespread adoption of such assistance systems the use of cost-efficient components is of crucial importance. Ultrasonic sensors fulfill these requirements on cost-efficiency. Consequently, they are widely used in the automotive industry for periphery surveillance in context of low velocities [7], [8], [9]. A prominent example is the meanwhile ubiquitous parking assistant, giving feedback on the distance to possible obstacles while the driver is backing into a parking lot. However, the sensitivity of ultrasonic sensors to external disturbances such as gusts of wind or rain and their restricted range [10], [11] was for a long time prohibiting in context of high-speed applications, such as the detection of cars in the blind spot of the driver. Another hindrance is the comparably low amount of information contained in the signal. In contrast to more expensive radar, lidar or camera systems [12], [13] that offer an acceptable angular resolution, us-sensors often have a wide aperture. That makes it difficult to distinguish the source or location of an echo.

The work at hand provides an insight into the development of a lane-change-assistent system using cost-efficient ultrasonic-sensors for the detection of objects in the driver's blind spot zone at absolute velocities of up to 160 km/h. A fuzzy-markov based approach using an inverse-geometric model has been proposed in [14] achieving promising detection rates. One hindrance is that this concept delivers no information about the type and velocity of the object in the blind spot zone. Another problem is the high number of underlying parameters that need to be tuned individually.

To tackle these issues, the work at hand investigates a model-based approach incorporating artifical neural networks. These networks [15], [16] are applied to diverse tasks like image analysis for traffic sign recognition in terms of driver-assistance or car identification. The outlined approach compares the ultrasonic signal with a precalculated curve data base for different situations in the blind spot zone like approaching cars, infrastructure or stationary objects. The correlation-coefficients are feeded into an artificial neural network and the trained network is used for the decision process.

This paper is organized as follows. In Section II the system-requirements and setup are discussed. Section III focuses on the design of the algorithm incorporating curve-fitting and neural networks. In Section IV the results are statistically

[1]Florian Mirus, Jürgen Pfadt, Christian Connette and Alexander Verl are with the Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Robot Systems, Nobelstraße 12, 70569 Stuttgart, Germany www.ipa.fraunhofer.de

[2]Björn Ewert and Dietmar Grüdl are with Valeo Schalter und Sensoren GmbH, Laiernstraße 12, 74321 Bietigheim-Bissingen, Germany www.valeo.com

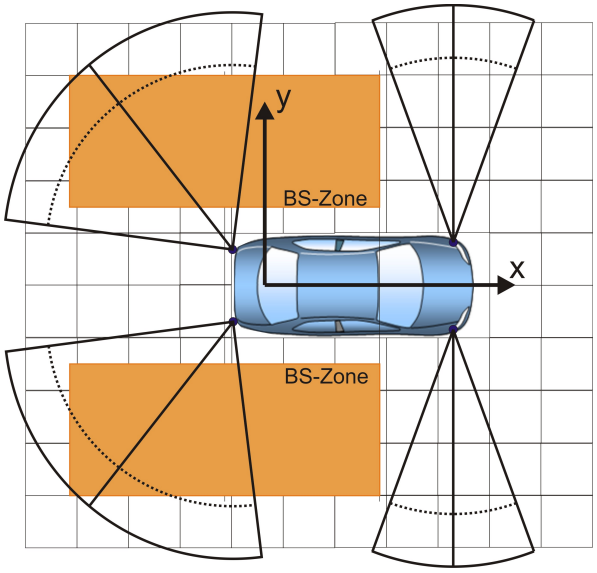


Fig. 2. Outline of the host car setup with ultrasonic sensor cones and orange blind spot zone; the grid is composed by 1 m to 1 m tiles

evaluated by comparing them to the requirements discussed in Section II. A little prospect on possible improvements concludes this paper.

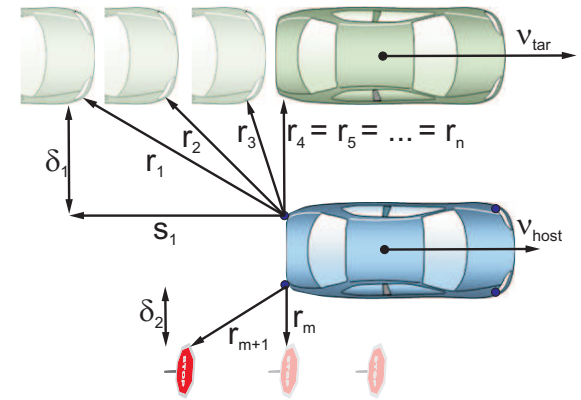
II. Problem Formulation

A. Preliminaries

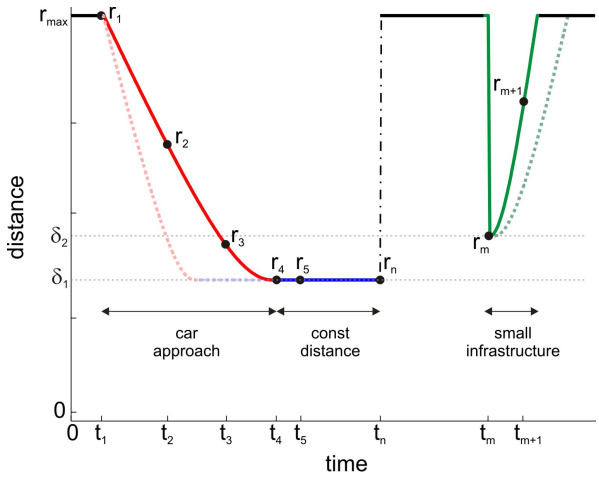
The goal of a blind spot surveillance system is to assist the driver when changing lanes and avoid possibly dangerous situations. The blind spot zone ranges from 3 m behind the car to the side mirrors and 3 m laterally. A warning signal shall be emitted when a car occurs within this zone during a lane change. To ensure applicability, some preconditions are necessary. For an optimal performance, a maximal reaction time of 300 ms is desired and the overall detection time should not exceed 1500 ms. A low false-alarm-rate is also required since too many missed or unnecessary warnings corrupt the driver's faith in the system's reliability. The operating range must be designed to detect blind spot alerts to a speed difference between the host and traffic from 0 to 30 km/h.

B. System Setup

The host vehicle is equipped with 12 ultrasonic sensors equally positioned at its front and its back side. To detect overtaking vehicles, the approach at hand evaluates the measurements of two sensors on each side of the host (dark black cones in Figure 2), namely the front and rear outer sensor. All other sensors are not used for blind spot detection. The aperture of the two rear sensors is approximately 75°, while the aperture of the front sensors is set to 45°. This enables sharp measurements with the front sensor in order to detect incoming traffic from the front or outgoing traffic from the back. In case of traffic residing within the blind spot zone, the driver is notified by illuminating a red light in its side mirror.



(a) Overtaking maneuver and pass of a traffic sign



(b) Corresponding sensor signals

Fig. 3. Correlation of driving situations and sensor signals

III. Realization

A. Curve-Fitting

The incoming ultrasonic measurements provide the minimal distance of the sensor to the object which reflected the ultrasonic beam in meters. Analysis of simulations as well as real road situations show that most overtaking maneuvers can be modelled by a parallel passing of two objects with constant orthogonal distance. This results ideally in a parabolic measurement signal.

Figure 3 illustrates the signal development for a car approaching from behind and the host passing a traffic sign. At timestamp t_1 the target enters the rear left sensor's range with measured distance r_1. The approach happens during t_2, t_3 with sensor measurements r_2, r_3. Then the target vehicle drives parallel to the host and the input signal fades to constant measured distances $r_4, \ldots, r_n$. When the target leaves the rear sensor's range, the signal rebounds to its maximum. Analogously, the measurements r_m and r_{m+1} describe a traffic sign passed by the host with an almost vertical line at timestamp t_m fading to an ascending parabolic function at t_{m+1}. Obviously, driving beside infrastructure like walls or side rails can be modelled by horizontal signal lines.

The dependance of the functions on the orthogonal distance

and the velocity is illustrated by the dotted lines in Figure 3(b). The orthogonal distance δ determines the maximal signal amplitude in vertical direction.
The velocity affects the signal structure as follows. A higher relative velocity $v_{rel} = v_{tar} - v_{host}$ causes a higher instantaneous rate of change in the signal curve discribing the minimal distance to the target vehicle which is denoted by the dotted red line in Figure 3(b). Analogously, a lower host speed v_{host} results in a slower rise of the distance to the traffic sign denoted by the dotted green line in Figure 3(b).
The functions used for modelling overtaking maneuvers have the form

$$f_i(t) = \sqrt{a_{i2}t^2 + a_{i1}t + a_{i0}} \tag{1}$$

with coefficients a_{ij} depending on the orthogonal distance δ and the velocity v_{rel} relative to the host speed to compare the signal with. Let

$$\delta_i \in \{0.5, 1, 1.5, \ldots, 4\} \text{ in m}, \tag{2}$$

$$v_i \in \{1, 3, 5, \ldots, 15\} \text{ in m/s} \tag{3}$$

and r_{max} be the maximal range of the sensor, the coefficients are calculated as follows: the sensor signal, i.e. the minimal distance of the sensor to the object, is modelled by the function $f_i(t)$, which shall be expressed in dependency on δ_i and v_i. As a start the pythagorean theorem is used to express

$$f_i(t)^2 = \delta_i^2 + s_i(t)^2 \tag{4}$$

in dependency on the orthogonal distance δ_i, the distance driven by the target vehicle

$$s_i(t) = s_{max} - v_i t \tag{5}$$

and the maximal parallel sensing distance

$$s_{max} = \sqrt{r_{max}^2 - \delta_i^2}. \tag{6}$$

By application of the equations 5 and 6, equation 4 is transformed to

$$f_i(t)^2 = v_i^2 t^2 - 2v_i\sqrt{r_{max}^2 - \delta_i^2}\,t + r_{max}^2, \tag{7}$$

so the coefficients in equation 1 are

$$a_{i0} = r_{max}^2, \qquad a_{i1} = -2v_i\sqrt{r_{max}^2 - \delta_i^2}, \tag{8}$$

$$a_{i2} = v_i^2. \tag{9}$$

Analogously the parabolic signals caused by stationary objects in the blind spot zone are modelled by functions $f_i(t)$. In this case the host vehicle's velocity v_{host} and the relative speed $v_{rel} = v_{host} - 0$ coincide. The descending part of the function can be neglected (see Figure 3). Infrastructure like walls or side rails is modelled by horizontal signal lines. Hence, the function data base contains 64 functions depending on δ_i and the relative velocity v_i for the detection of approaching cars, 8 functions depending on δ_i and the host speed v_{host} for the detection of small stationary objects and horizontal lines simulating walls or side rails depending on the sample mean of the measurements.
A moving window containing n sequenced measurements of size $n = 8$ for short term and $n = 32$ for long term analysis is considered. Let $W_m = \{x_1, \ldots, x_n\}$ be the set of the incoming data. For better results $2n$ function values $W_{f_i} = \{y_{i1}, \ldots, y_{i(2n)}\}$ are calculated and all subsets $\{y_{i(1+k)}, \ldots, y_{i(n+k)}\}$ for $k = 0, 1, \ldots, n$ containing n sequenced elements of W_{f_i} are compared wih W_m. For some curves additional modifications like considering only the relevant function values (i.e. ignoring too many subsequent constant values) and shifting them to the center of the window in order to improve the detection are made. The first algorithmic step is the choice of an adequate f_i satisfying

$$\min_{f_i,k}\{\sum_{\substack{j=1 \\ j\neq j_{max}}}^{n} \left|x_j - y_{i(j+k)}\right|\} =: F(W_m, f_i) \tag{10}$$

with

$$j_{max} = \max_j\{\left|x_j - y_{i(j+k)}\right|\}. \tag{11}$$

From the coefficients of the chosen function f the target vehicle's orthogonal distance to the host and relative speed can be recalculated as follows

$$v = \sqrt{a_2}, \qquad \delta = \sqrt{r_{max}^2 - \frac{a_1^2}{4a_2}}. \tag{12}$$

Along with the characteristic values sample mean and covariance in every sensor's moving window W_m the calculated distance δ and relative velocity v form the input data of the neural network.

B. Neural Networks

1) Design: Since the curve data base is calculated over a lattice of orthogonal distance and relative velocity, it is impossible to detect a unique fitting function in most cases. As illustratd in Figure 4, there are several functions with similar deviation values. An artificial neural network is able to tackle this problem and refine the decision process.
In this paper a feedforward neural network, which means that there are no cycles within, containing twenty neurons within the hidden layer is used. The training was realized by supervised learning using the recorded data from different test drives as input values for the Levenberg-Marquardt-Algorithm.
This input values are the sample mean and covariance of all sensors in the current moving window W_m, the results of the curve-fitting-process represented by a state variable indicating whether an approaching car, a stationary object, constant distance or none of those cases has been detected and the sum of the detected states covering the last second of measurements. Additionally, the deviation function $F(W_m, f)$ of the best fitting function f for every state and the host vehicle's velocity is entered. The approach at hand is illustrated in Figure 5.
The neural network's binary output value is 1 if a possibly dangerous state has been detected, 0 otherwise. A warning is emitted, if two of the last three output values are nonzero.

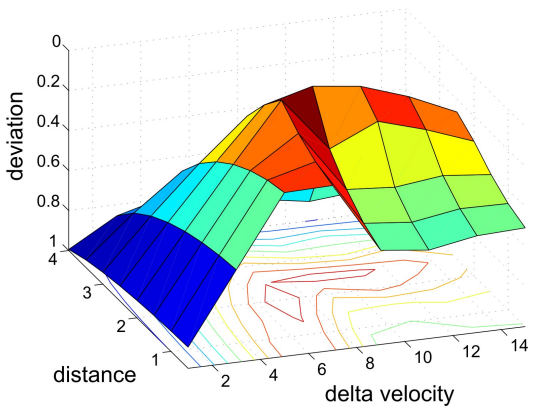


Fig. 4. Deviation function over the distance-velocity-lattice

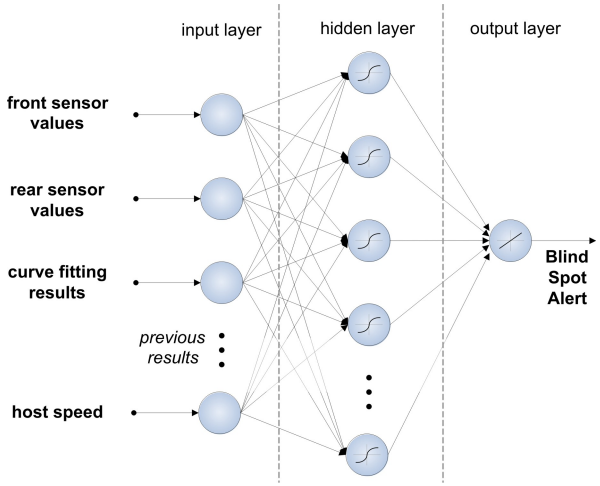


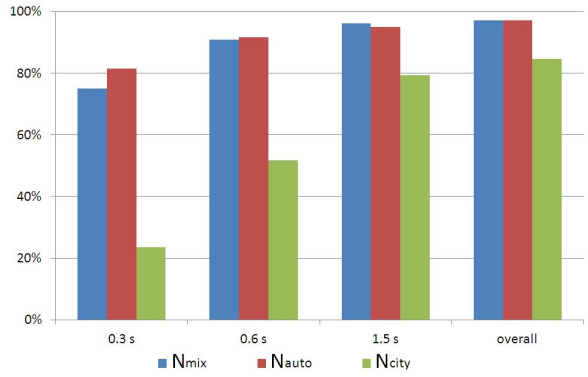
Fig. 5. Neural network for the detection of overtaking cars

2) Training: There are several ways to realize this neural network approach. One possibility is to train different networks for each environment namely inner city, interurban and motorway traffic. Another idea is to design some kind of one-size-fits-all network, whose training set contains a mix of test drives in different environments.
The work at hand illustrates both possibilities demonstrating three neural networks N_{auto},N_{city} and N_{mix}. The training set of N_{auto} contains three motorway files with 45 km driven distances and 88 overtaking maneuvers, N_{city} was trained using two inner city files with 10 km driven distance and 46 overtaking maneuvers and the underlying training data of N_{mix} is a combination of these motorway and inner city test drives in a ratio of 3:2 with 55 km driven distance and 134 overtaking maneuvers. In this first attempt interurban drives have been left out of the training files since the chosen networks are expected to cover that cases at a satisfying level.

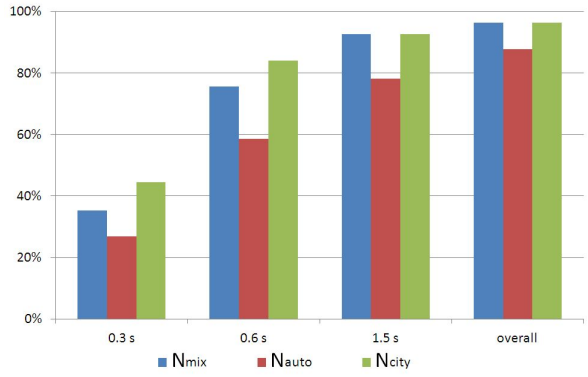
IV. Results

A. Statistical Evaluation

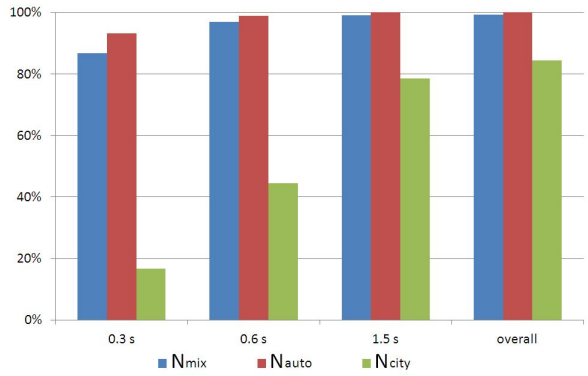
To revise the functionality and perfomance of the proposed procedure, extensive testing has been conducted. The host vehicle was equipped with one laser sensor on each side and four color cameras mounted on top of the car to generate a 360° view of the environment. In order to ensure meaningful results, differing types of target vehicles like cars, motorbikes or trucks and different road environments like inner city, interurban or motorway drives had to be considered. After more than 2000 km of test drives, the data base contains over 3000 test cases for qualitative and quantitative evaluation.
The three networks N_{auto},N_{city} and N_{mix} have been applied to a collection of test files containing approximately 356 km driven on motorways and addionally about 20 km and 32 km driven in city respectively interurban traffic.
According to the requirements stated in Section II-A the



(a) Detection rates without host speed limits



(b) Detection rates within 25 km/h to 50 km/h (moderate speed)



(c) Detection rates within high-speed-interval (over 70 km/h)

Fig. 6. Statistical evaluation of detection rates for different host speed intervals

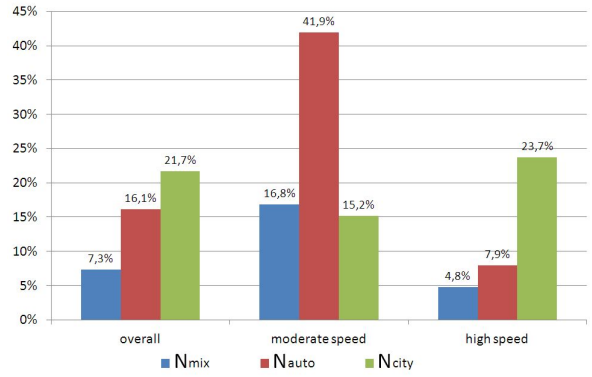


Fig. 7. False-alarm-rate for all networks and different host speed intervals

desired maxmial reaction time shall not exeed 300 ms. Hence detection intevals of 0.3 s, 0.6 s, 1.5 sand the overall detection rate without any time limit are statistically evaluated. The performance of the networks is compared in three settings with different host speed intervals to illustrate their particular strengths. Figure 6 illustrates the detection rates for the different speed-intervals. The first setting, illustrated in Figure 6(a), has no limitations concerning the host vehicle's speed to illustrate the overall performance of all networks. Figure 6(b) shows the results in the second interval (moderate-speed-interval) ranging from 25 km/h to 50 km/h to evaluate the inner city efficiency. Finally the minimal host speed of the third interval (high-speed-interval, see Figure 6(c)) is set to 70 km/h. Figure 7 shows the false-alarm-rate of all networks relative to the total number of emitted warnings.

B. Discussion

The results demonstrated in Figure 6 and Figure 7 show that the networks N_{auto} and N_{city} achieve promising detection rates for the particular driving situations they have been trained for. As expected N_{city} performs best in terms of moderate velocities where a slightly elevated reaction time is acceptable achieving an overall detection rate of 96.3% and even 84% within 0.6 s. Analogously, N_{auto} provides satisfying detection rates in terms of high velocities detecting all vehicles and even 93.1% within 0.3 s. The one-size-fits-all network N_{mix} provides low false-alarm-rates in the overall and high-speed setting in exchange for a slightly elevated reaction time but still achieving overall detection rates of at least 96.3% in every setting.

C. Prospect

As a start the results of the neural network approach at hand show promise. Since every network has its strengths in particular situations, there are several possibilities for future investigations. Although the one-size-fits-all network provides a solid overall performance the training of different networks for several situations is prefered since the specialized networks provide even better detection rates for their particular strengths within less reaction time. Since it is possible to detect the actual traffic situation via odometry and curvature information, a deeper analysis of this approach is intended.

Another aspect demanding further investigations is a neural network trained for rain weather conditions including wet roads and splash water. In this case, the ultrasonic sensor signal contains a lot of noise, so it might be necessary to consider alternative reference functions.

References

[1] S. Thrun, "Winning the darpa grand challenge," in *Knowledge Discovery in Databases: PKDD 2006*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Springer Berlin / Heidelberg, 2006, vol. 4213, pp. 4–4, 10.1007/11871637-4. [Online]. Available: http://dx.doi.org/10.1007/11871637-4

[2] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, pp. 123–139, 2009, 10.1007/s10514-009-9115-1. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9115-1

[3] K. Macek, D. Vasquez, T. Fraichard, and R. Siegwart, "Safe vehicle navigation in dynamic urban scenarios," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, oct. 2008, pp. 482 –489.

[4] S. Kolski, D. Furgeson, C. Stachniss, and R. Siegwart, "Autonomous driving in dynamic environments," in *Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[5] M. Ye, J. Wu, and Y. Xiu, "Research of driver assistance system based on pervasive computing," in *Computer-Aided Industrial Design Conceptual Design, 2009. CAID CD 2009. IEEE 10th International Conference on*, Nov. 2009, pp. 2342 –2344.

[6] M. Ma, J. Chen, and G. Xu, "Research on adaptive road control algorithms of anti-lock braking system," in *Computer-Aided Industrial Design Conceptual Design, 2009. CAID CD 2009. IEEE 10th International Conference on*, Nov. 2009, pp. 2238 –2240.

[7] W.-J. Park, B.-S. Kim, D.-E. Seo, D.-S. Kim, and K.-H. Lee, "Parking space detection using ultrasonic sensor in parking assistance system," in *Intelligent Vehicles Symposium, 2008 IEEE*, June 2008, pp. 1039 –1044.

[8] C. Gearhart, A. Herold, B. Self, C. Birdsong, and L. Slivovsky, "Use of ultrasonic sensors in the development of an electronic travel aid," in *Sensors Applications Symposium, 2009. SAS 2009. IEEE*, Feb. 2009, pp. 275 –280.

[9] X.-T. Le, S.-B. Oh, W.-S. Lee, C.-S. No, and S.-H. Han, "Design of intelligent mobile robot system based on ultrasonic sensors," in *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, Oct. 2007, pp. 72 –76.

[10] V. Magori, "Ultrasonic sensors in air," in *'Proc. IEEE Int. Ultrasonics Symp. (Cannes)'*, 1994, pp. 471–81.

[11] ——, "Signal processing for smart ultrasonic sensors," in *CompEuro '89., 'VLSI and Computer Peripherals. VLSI and Microelectronic Applications in Intelligent Peripherals and their Interconnection Networks', Proceedings.*, May 1989, pp. 3/21 –3/26.

[12] J. Wang, G. Bebis, and R. Miller, "Overtaking vehicle detection using dynamic and quasi-static background modeling," in *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, June 2005, p. 64.

[13] P. Batavia, D. Pomerleau, and C. Thorpe, "Overtaking vehicle detection using implicit optical flow," in *Intelligent Transportation System, 1997. ITSC '97., IEEE Conference on*, Nov 1997, pp. 729 –734.

[14] C. Connette, J. Fischer, B. Maidel, F. Mirus, S. Nilsson, K. Pfeiffer, A. Verl, A. Durbec, B. Ewert, T. Haar, and D. Grüdl, "Rapid detection of fast objects in highly dynamic outdoor environments using cost-efficient sensors," *accepted for the ROBOTIK 2012 conference in Munich*, 2012.

[15] C. Goerick, D. Noll, and M. Werner, "Artificial neural networks in real-time car detection and tracking applications," *Pattern Recognition Letters*, vol. 17, no. 4, pp. 335–343, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167865595001298

[16] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. Seelen, "An image processing system for driver assistance," *Image and Vision Computing*, vol. 18, no. 5, pp. 367–376, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885699000323

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

ESTRO: Design and Development of Intelligent Autonomous Vehicle for Shuttle Service in the ETRI

Jaemin Byun, Ki-In Na, Myungchan Noh, Joochan Sohn and Sunghoon Kim

Abstract— **ESTRO(ETRI Smart Transport RObot) project aims at the development of autonomous vehicle to transport goods and people without the help of driver in the well-structured area such as campus. The autonomous vehicle, ESTRO has been designed and implemented by modifying electronic vehicle. In addition, the cost of sensors and the complexity of system are minimized on the purposed of a commercial autonomous driving system in urban traffic environment. This paper proposes the design of H/W and S/W architecture for the autonomous vehicle and describes the method of environmental perception and navigation. The implemented system has been tested in ETRI campus.**

I. INTRODUCTION

Through the technologies of autonomous driving have developed, it is possible to drive safely and conveniently in complex environment with dynamic objects such as vehicles and pedestrians.

Recently, autonomous vehicle has a lot of problems on the legal and technological issue for commercialization, so most of main technologies have been just applied for ADAS (Advanced Driver Assistance System) products until now. The Google driverless cars have officially licensed in Nevada, these vehicles are being tested around real traffic environment on the state [1]. The Stadtpilot project' autonomous vehicle (Leonie) has shown to the ability of driving autonomously in real traffic environment of Braschchweig, Germany [2]. However, in the Republic of Korea, there is no legal framework which enables autonomous driving on public roads.

Therefore, ESTRO project aims at autonomous driving with low-speed in the well-structured section such campus and area where the specialized traffic regulations are applied. ESTRO system has developed as a robotic vehicle for transporting supplies and carrying people to final destination without driver's support. The ESTRO can perform the call service that user can call the autonomous vehicle to user's requested place with mobile devices using wireless communication.

This work was supported by the ETRI Research and Development Support Program of MKE/KEIT

Jaemin Byun is with the Electronics and Telecommunications Research Institute, Daejeon, Korea

(corresponding author to provide phone: +82-42-860-6545; e-mail: jaemin.byun@etri.re.kr.).

Ki-In Na, Myungchan Roh, Joochan Sohn and Sunghoon kim are with the ETRI, Daejeon, Korea, (e-mail: {kina4147, mcroh, jcsohn, saint}@etri.re.kr).

A. Relative Works

Important events for the autonomous vehicle research are DARPA Grand Challenges and Urban Challenge. The Grand Challenges in 2004 and 2005 were held in the Mojave Desert, America. The objective of Grand Challenges was to create the first fully autonomous ground vehicle capable of completing a substantial off-road course within a limited time. There was no winner at the first Grand Challenge, but five vehicles successfully completed the race at the second Grand Challenge. The Urban Challenge in 2007 took place for further advanced vehicle requirements to include autonomous operation in the urban environment. In this competition, the six teams were successfully finished the given course. Mainly, the vehicles of Stanford University and Carnegie Mellon University are well operated in both the second Grand Challenge and the Urban Challenge. Both Junior of Stanford University and Boss of Carnegie Mellon University had Applanix POS-LV220/420, Velodyne HDL-64 3D LIDAR, IBEO Alasca XT LIDAR, RADAR and cameras. These vehicles mainly perceived surrounding information with LIDAR and continuously detected its position with GPS/INS equipment [3], [4]. This configuration for the autonomous vehicle has become common after these competitions. Furthermore, the autonomous vehicle has been researched much actively.

Europe countries and America are actively researching and developing the autonomous vehicle. INRIA, France has been developing the robust electric autonomous vehicle, the Cybercar using 2D LRF-based SLAM and V2V/V2I communication [5]. In 2010, VisLab ran the VisLab Intercontinental Autonomous Challenge, a 15,000km test of autonomous vehicles from Parma, Italy to Shanghai, China [6]. Moreover, Autonomous Labs of Freie University, Germany has been developing the autonomous vehicles with 3D LIDAR and cameras [7]. This team also has succeeded the test autonomous driving in Berlin's street and highways in 2011. MuCar-3 with the 3D LIDAR is being developed by university of the Bundeswehr Munich, Germany [8], [9]. This project mainly is focused on the LIDAR-based 3D object perception. Google have been developing fully autonomous vehicle, Google Driverless Car, equipped with cameras inside the car, a 3D LIDAR on top of the vehicle, RADAR on the front of the vehicle and a position sensor attached to one of the rear wheels that helps locate the car's position on the map. This project is currently famous in autonomous vehicle research and is being led by Google engineer, Sebastian Thrun who is also director of the Stanford Artificial Intelligence Laboratory which developed both Stanley and Junior [10], [3].

B. *Outline*

Section II describes the platform and the software architecture of the developed autonomous vehicle, ESTRO. In section III, the method for environmental perception will be described such as on-road marker detection with cameras, curb and obstacle detection with LRFs, and localization with GPS, odometer and on-road marker. Moreover, local map, the integration form of multiple sensory data will be also introduced. In section IV, the behavior planning, the path planning and its control will be introduced. Experimental scenarios such as normal road, intersection and parking lot will be demonstrated and discussed in section V. Finally, Section VI closes with conclusions.

II. VEHICLE PLATFORM & SOFTWARE ARCHITECTURE

A. *Vehicle Platform*

ESTRO has been being developed since 2008 at ETRI. The objective of this autonomous vehicle is the unmanned shuttle system which can autonomously transfer human and load to everywhere in ETRI. It includes two LRFs; one is equipped on the top of the vehicle for extracting curb and the other is equipped at the front of the vehicle for detecting obstacles. Three CCD cameras are also used for detecting on-road markers such as lane, crosswalk, speed bump, and stop line. The GPS on the vehicle and the odometer at rear wheel were arranged for localization. Touch screen monitor and speakers are set for communication with users as shown in Fig. 1.

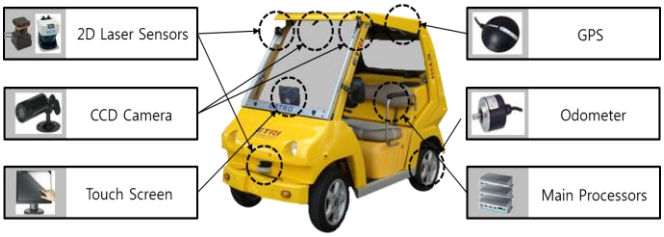


Figure 1. ESTRO hardware configuration

B. *Software Architecture*

The software architecture for the autonomous vehicle system has to be designed efficiently because the autonomous system is too complex and huge to operate in real-time and to understand its structure easily. ESTRO also has various types of devices and various components have to be separately executed at the same time. Therefore, the software architecture for ESTRO has also efficiently designed as shown in Fig. 2.a. The designed software architecture for ESTRO has four module; perception module, navigation module, GUI module, and system monitoring module.

Perception module perceives environmental information such as on-road markers, curb, obstacles, and current position with cameras, LRFs, GPS, and odometer. It also builds the local map, which various types of sensor data were integrated into. Navigation module gets environmental information in the form of the local map from perception module and performs both behavior planning and path planning. In addition, it can also generate the directional commands to control the autonomous vehicle continuously for following the planned path. GUI module shows the current condition of the ESTRO periodically and transfers user commands to the vehicle operation system. The system monitoring module always monitors faults of the operating components and keeps them running safely. The designed software architecture for ESTRO is developed using OPRoS (Open Platform for Robotic Service) components [11]. According to the functions of component, components are distributed into each module and components in module consist of atomic components or composite components consisting of atomic components as Fig. 2.b.

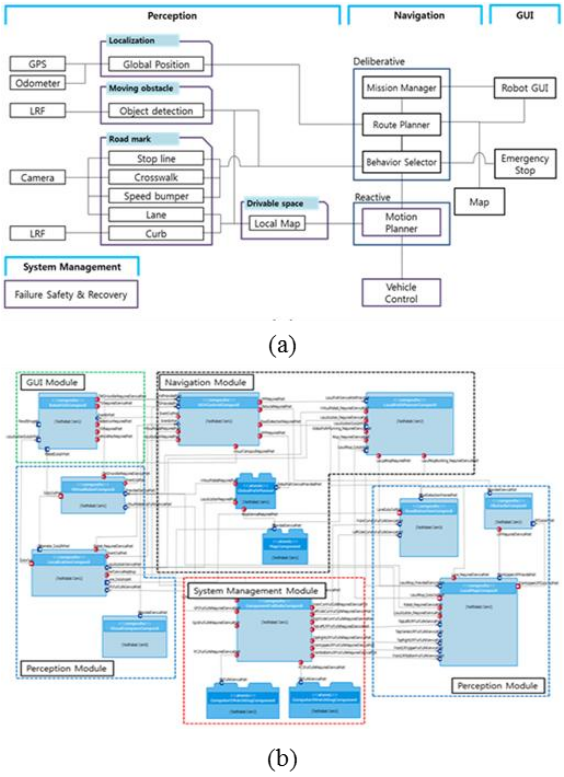


Figure 2. Software architecture of ESTRO; it consists of perception module, navigation module, GUI module and system monitoring module

III. ENVIRONMENTAL PERCEPTION

For the environmental perception, ESTRO has various types of sensors such as three cameras, two LRFs, odometer, and GPS. The surrounding information on the road such as curb, obstacle, on-road markers and position are detected from each sensor component. All the acquired data from sensor components are integrated and displayed in the form of the local map, which is the occupancy grid map including surrounding information for navigation as shown in Fig. 3.

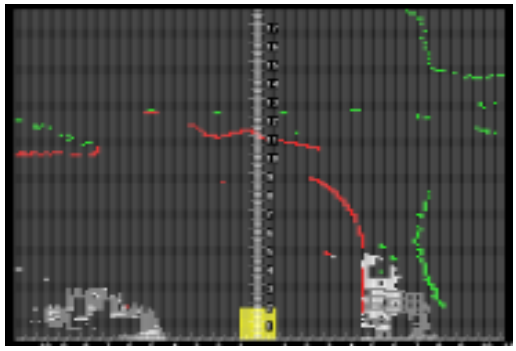


Figure 3. The generated local map

A. *Road Recognition and Obstacle Detection*

Before collecting sensor data from cameras and LRFs equipped on ESTRO, intrinsic and extrinsic calibrations are performed with a planar checkerboard pattern [12]. After solving for constraints between the views of a planar checkerboard calibration pattern from cameras and LRFs, their coordinate systems are calibrated to the same vehicle coordinate system.

For on-road markers detection, a raw colored image is converted into a gray scale image at first. Adaptive rectangular ROI (Region of Interest) extraction and noise filtering is also performed. Next, edge extraction through sobel approach and line fitting through Hough transformation method are achieved to get characteristics of the extracted lane. Speed bumper, crosswalk and stop line are also detected in the similar way to lane detection as shown in Fig. 4 [13].

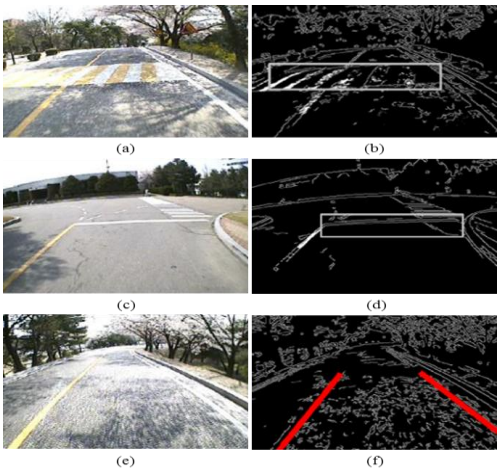


Figure 4. On-road markers extraction; (a) original image of normal road, (b) extraction for speed bumper, (c) original image of stop line, (d) extraction for stop line, (e) orginal image of lane, (f) extraction for lane

The LRF on the top of the vehicle is used for detecting curb, which is the raised edge of a pavement or sidewalk. Firstly, curb shape is geometrically recognized and its position is also derived with LRF data. Secondly, the position of curb is estimated and is also tracked using particle filter approach [14].

Obstacles on road are detected by the LRF at the front of the vehicle, which is arranged in the parallel with ground as shown in Fig. 1. All the detected obstacles are segmented and its size and distance are also estimated [15]

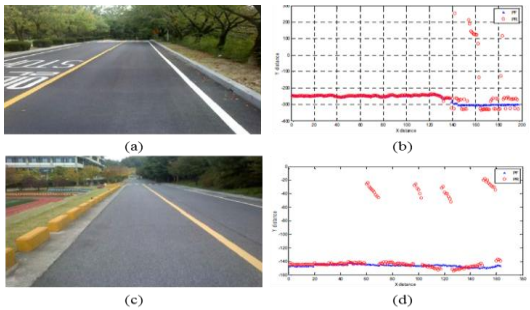


Figure 5. Curb extraction; (a) image of continuous curb, (b) extraction for continuous curb, (c) image of discontinuous curbs, and (d) extraction for discontinuous curbs.

B. *Localization*

Firstly, the current position of ESTRO is continuously calculated using GPS and odometer. This derived position value contains some error. However, ESTRO is assumed to be operated at well-known roads such as ETRI campus where on-road marker information such as lane and stop line is already stored in the map. Therefore, both the lateral and the longitudinal distance error in the position calculated by GPS and odometer can be compensated using on-road marker with Extended Kalman Filter as shown in Fig. 6. Besides, for reducing the sensors error such as drift error and jumping position of GPS, Mahalanobis distance approach is also applied. As a result, the accuracy of the estimated position is better than normal EKF localization [16], [17].

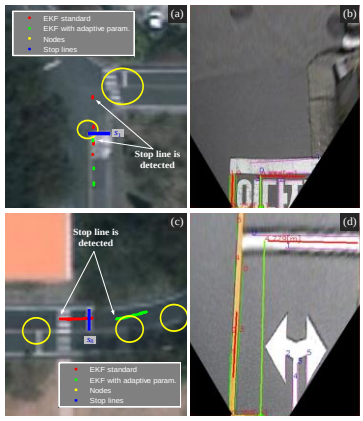


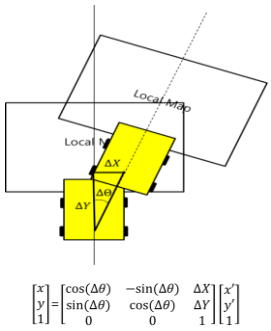
Figure 6. Comparison of two position estimates (EKF with and without adaptive parameter) near the stop line. (a) and (c) are the robot positions when the robot is detecting the stop lines. (b) and (d) are the bird-view images of mono-camera when the robot is detecting the stop line.

C. *Local Map Building*

For integrating multiple data from various sensors, local map is applied. On-road markers such as lane, speed bump, crosswalk, and stop line from cameras are described as typical representative values. For example, lane can be represented by its starting point and slope. In addition, crosswalk and speed bump are represented its distance and size. After transmitting these transformed data to the local map building component, they are integrated into the local map altogether. The detected curb and obstacle information are also described as relative position and size by LRF component and are also displayed in local map.

The local map has to be continuously updated, because the vehicle is moving. To update the previous local map, both relative pose and position change of the vehicle has to be periodically measured such as rotation angle and translation value. The transformed previous local map is integrated to current local map which include only current sensory data as shown in Fig. 7. In other words, local map contains both previous and current surrounding information at the same time.

In the local map, the position of the vehicle is fixed at bottom and middle of the map as shown in Fig. 3. The derived current position from localization component is matched with the fixed vehicle position in local map. In addition, the other positions of local map also are derived based on this position connectivity relatively.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & \Delta X \\ \sin(\Delta\theta) & \cos(\Delta\theta) & \Delta Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Figure 7. Local map transformation and update

IV. Planning And Control

A. Behavior Planning

ESTRO can select proper behaviors corresponding to the changes of road environment for driving safely and efficiently. Most of real road environments are composed of normal road which has well-painted lane, intersection, speed bump, cross walk, etc. According to road environment, the vehicle should choose its proper driving mode. For example, the basic behavior mode, normal driving mode, is to keep the certain distance between the vehicle and well-painted lane on the road environment.

Moreover, the vehicle also performs obstacle avoidance by reducing the speed of the vehicle and avoiding obstacles when they suddenly appear in front of vehicle. Suitable states have to be selected according to input information and it also transits to another suitable state through proper surrounding information shown in the Fig. 8.a. The state transition diagram is designed by analyzing the pattern of driver's behavior.

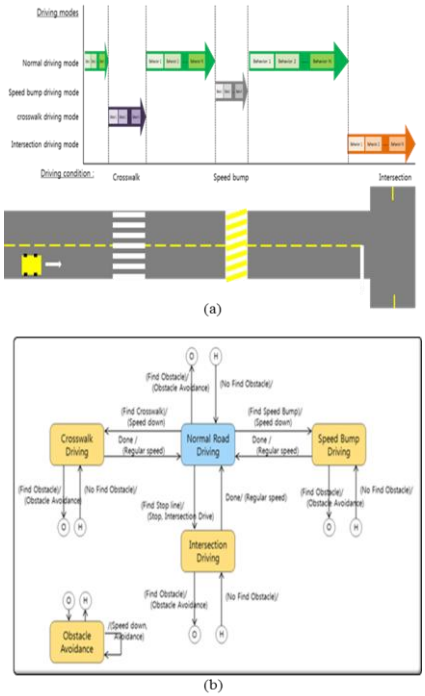


Figure 8. Behavioral planning; (a) Scenario of the driving control system depending on driving condition, driving mode and behaviors, and (b) Diagram of driving condition transfer

B. Path Planning and Control

To reach the desired destination by autonomous driving, it is essential to include both the global and the local path planning. Therefore, the ESTRO system is largely separated into two steps of path planning. First step is global path planning which generates routes to pass and to reach for final destination. The global path planer performs path planning with the topological map information includes in the road connection relation and physical distance among neighbor nodes. Furthermore, the optimal path is generated by minimizing cost function which means the total traveling distance based on Dijkstra algorithm. The results of it are information on list of the node included in road property and the relation among nodes, while it traveling from start point to final destination. Next step is local path planning which performs periodically according to change of environment, it is decided the way by the result of above introduced behavior planning where the vehicles drives on the normal road or free form road such as intersection and parking lot area.

The implemented local path generator is based on three degree of Bezier curve [18]. The planned path could be smooth enough for ESTRO which is car-like model to track and to follow it. The important step for deciding the shape of Bezier curve is to pick up control points. By considering of processing time and complexity, the ESTRO system is based on three degree of Bezier curve as shown in Fig. 9.

For the three degree of Bezier curve, the four points have to be decided as control points in the normal case. The first point means the current position of the vehicle and last point means the position of next node which is decide by global path planning. A lot of candidate paths are generated at the same time by changing the position of rest two points on the center line of the current road. To get an optimal path among a lot of generated paths, every path is evaluated with three criteria such as kinematics constraint, obstacle collision, degree of smoothness. Finally, the optimal path can be selected, which has low cost.

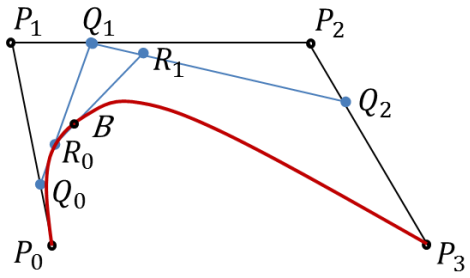


Figure 9. Beizir curve of degree 3 at t=0.5.

In the case of free-from road such as intersection and parking lot, the path planning module generates Bezier curve from the start configuration, $q_s = (x_s, y_s, \theta_s)$ to the target configuration $q_t = (x_t, y_t, \theta_t)$. The feasible paths can be generated by changing the second control point P_1 and the third control point P_2 as shown in Fig. 9. The first control point P_o and the fourth point P_3 are located at the start and the target node. For considering the various positions of the second control point, they are propagated the constant along the line with θ_s slope. The third control point is accomplished with the same procedure. Therefore, the different paths for a target state

can be generated. As a result, the optimal path can be determined among candidate paths by evaluating and comparing cost of paths. To follow accurately the generated optimal path above, the pure pursuit method is applied. The pure pursuit method generates steering angle and velocity of vehicle [19]. The important factor of this method is a look-ahead distance. The look-ahead distance can be decided based on prior knowledge of the road on the map. With this method, the lateral tracking error of ESTRO in the test site is fewer than 50cm.

V. Experiment and Result

To show the performance of the developed autonomous vehicle, ESTRO, the real driving test was performed on 7 km road environment of ETRI including many possible traffic situations and various types of roads such as well-structured road, intersection, parking lot, etc. as shown in Fig. 10. In this test site, ESTRO conducted various types of driving including lane keeping, speed control, obstacle avoiding, intersection driving, etc. as shown in Fig. 11.

Figure 10. The map of ETRI campus(more than 7 km real road environment)

Figure 11. (a) ESTRO stopped in front of stop line for compensation on the intersection, and (b) ESTRO stopped when the pedestrian crossed the road.

A. Normal Road

Most of roads in the test site are well-structured road which has both lanes and curbs on one side or on both sides as Fig. 12. However, most of them are surrounded by trees and buildings, so it is not easy to get high accuracy position with the equipped low-cost GPS (over RMS 2m on average). Furthermore, it is impossible to drive autonomously depending on only localization information. Thus, the vehicle has to compensate position derived by GPS and odometer with pre-saved road information in the digital map such as lane, stop line, etc.

The mid-point of current road is calculated by recognized curb and lanes and pre-saved road information such as the road width, the number of lanes, etc. The vehicle can drive by following the calculated midpoint. Speed of the vehicle is about 10 ~ 20 km/h. Average tracking error which is difference with mid-point of road is less than 30cm.

Figure 12. Well-structured noraml road which has lanes and curbs

B. Intersection

Autonomous driving highly depends on the accuracy of location in the area of intersection without lanes as show in Fig. 13. Therefore, before the vehicle enters the intersection, position was compensated by left and right side lane and stop line information to improve the position accuracy. For this compensation, the vehicle has to stop for a short period when the distance between the front of vehicle and stop line is within 1m. The vehicle went forward if there are not obstacles such as pedestrians and vehicles are on the generated route. When obstacles appear, the vehicle stopped and started again to follow the planned path on intersection after obstacles disappeared.

Figure 13. The exmpale of intersection area in our test site

C. Parking Lot Area

As shown in Fig. 14, the driving in parking lot is based on the extraction of traversable area with the local map. The vehicle generates the virtual path to the next node on the traversable area of local map and generate steering angle for tracking the generated path. In parking lot, obstacle detection is important because the parked vehicle can be suddenly moved and pedestrian can appear.

Figure 14. The exmpale of parking lot area in our test site

VI. CONCLUSION

This paper explained about ESTRO project which aims at the development of autonomous vehicle to transport goods and people without the help of driver in the well-structured area such as campus. At the first, H/W configuration and S/W architecture of ESTRO were introduced. The methods for road recognition, obstacle detection, localization, and local map building for environmental perception were described. In addition, the methods for behavior planning, path planning, and control also were explained for planning and control. For demonstration of the developed vehicle, real driving test in ETRI campus was performed at normal road, intersection and parking lot.

ACKNOWLEDGMENT

This work was supported by the ETRI Research and Development Support Program of MKE/KEIT. [KI0041417, Development of Decision Making/Control Technology of Vehicle/Driver Cooperative Autonomous Driving System (Co-Pilot) Based on ICT].

REFERENCES

[1] J. Markoff. (2010, October) Google cars drive themselves, in traffic. Online. The New York Times. [Online]. Available: http://www.nytimes.com/2010/10/10/science/10google.html

[2] T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J. R. Bohmer, "Stadtpilot: First Fully Autonomous Test Drives in Urban Traffic," in *Proc. IEEE Int. Conf. Intelligent Transportation Systems*, Washington, USA, 2011, pp. 919-924

[3] M. Montemerlo et al., "Junior: The Stanford entry in the Urban Challenge," Journal of Field Robotics, vol. 25, no. 9, pp. 569–597, 2008.

[4] C. Urmson et al., "Autonomous driving in urban environments: Boss and the Urban Challenge," Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[5] V. Milanes, J. Alonso, L. Bouraoui, and J. Ploeg, "Cooperative Maneuvering in Close Environments Among Cybercars and Dual-Mode Cars," *IEEE Trans. Intelligent Transportation Systems, vol. 12, no. 1, pp. 15-24*

[6] A. Broggi, L. Bombini, S. Cattani, P. Cerri, and R. Fedriga, "Sensing requirements for a 13,000 km intercontinental autonomous drive," in Intelligent Vehicles Symposium (IV), 2010 IEEE, June 2010, pp. 500–505.

[7] R. Rojas, et al., "Spirit of Berlin: An Autonomous Car for the DARPA Urban Challenge - Hardware and Software Architecture," tech. rep., Free University of Berlin, 2007

[8] F. Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H. Wuensche., "Driving with Tentacles:Integral Structures for Sensing and Motion," Journal of Field Robotics, vol. 25, no. 9, pp. 640–673, 2008.

[9] M. Himmelsbach. "Real-time object classification in 3D point clouds using point feature histograms," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems,St. Louis, USA, 2009, pp. 992-1000*

[10] S. Thrun, et al., "Stanley: The Robot That Won the DARPA Grand Challenge," Springer Tracts in Advanced Robotics, vol. 36, pp. 1-43, 2007

[11] C. Jang, S. I. Lee, S. W. Jung, B. Song, R. Kim, S. Kim, and C. H. Lee, "OPRoS : A New Component-Based Robot Software Platform," ETRI Journal, vol. 32, no. 5, pp. 646-656.

[12] Q. Zhang and R. Pless," Extrinsic Calibration of a Camera and Laser Range Finder (improves camera calibration)," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 2301-2306

[13] J. Byun, J. Sung, M. C. Roh, and S. H. Kim,"Efficient and Robust Road Recognition for Autonomous Navigation in Structured Urban Environment," in *Proc. Int. Conf. Ubiquitous Robots and Ambient Intelligent,Incheon*, Repulic of Korea, 2010, pp. 273-277

[14] J. Byun, J. Sung, M. C. Roh, and S. H. Kim," Autonomous Driving through Curb Detection and Tracking," in *Proc. Int. Conf. Ubiquitous Robots and Ambient Intelligent,Incheon*, Repulic of Korea, 2011, pp. 273-277

[15] R. MacLachlan,, "Tracking Moving Objects From a Moving Vehicle Using a Laser Scanner," Carnegie Mellon University, 2005

[16] Christiand, Y. C. Lee, and W. Yu," EKF Localization with Lateral Distance Information for Mobile Robots in Urban Environments," in *Proc. Int. Conf. Ubiquitous Robots and Ambient Intelligent,Incheon*, Repulic of Korea, 2011, pp. 281-286

[17] Y. C. Lee, Christiand, W. Yu, and J. I. Cho, Adaptive Localization for Mobile Robots in Urban Environments Using Low-Cost Sensors and Enhanced Topological Map," in *Proc. Int. Conf. Advanced Robotics, Tallinn*, Estonia, 2011, pp. 569-575

[18] Ji-wung Choi, Renwick Curry, Gabriel Elkaim," Path Planning based on Bezier Curve for Autonomous Ground Vehicles,"in Proceeding of World Congress on Engineering and Computer Science, 2008

[19] Sunglok Choi, JaeYeong Lee, and Wonpil Yu," Comparison between Position and Posture Recovery in Path Following," " in Proceeding of Ubiquitous Robots and Ambient Intelligence (URAI), 2009

An effective 6DoF motion model for 3D-6DoF Monte Carlo Localization

A. L. Ballardini[1] A. Fúrlan[1] A. Galbiati[1] M. Matteucci[2] F. Sacchi[1] D. G. Sorrenti[1]

***Abstract*— This paper deals with the probabilistic 6DoF motion model of a wheeled road vehicle. It allows to correctly model the error introduced by dead reckoning. Furthermore, to stress the importance of an appropriate motion model, i.e., that different models are not equally good, we show that another model, which was previously developed, does not allow a correct representation of the uncertainty, therefore misguiding 3D-6DoF Monte Carlo Localization. We also present some field experiments to demonstrate that our model allow a consistent determination of the 6DoF vehicle pose.**

I. INTRODUCTION

In urban settings, autonomous driving is more similar to mobile robotics, because of the need to have a global localization of the vehicle. Localization cannot be managed using purely dead reckoning, e.g., wheel based odometry [1] [2] [3]. Wheel sliding, e.g., due to contact with the ground surface, weather conditions, unexpected values of the wheels diameters, etc., require the use of external sensors and the corresponding algorithms, to determine the vehicle position [2]. It must be noticed that in urban environments the GPS system, apparently an immediately available solution, has an absolutely not adequate reliability, with respect to the localization and navigation requirements, due to the frequent lack of signal [4] [5].

While the state of the art provides different solutions for the 2D - 3DoF localization problem, these solutions are primarily designed for indoor robotic environments, where the analysis of the motion in a 3D space can be simplified, favoring an estimation of the robot pose limited to a 3DoF pose in the 2D plane. 3D approaches known in the literature, e.g., [4], [5], [6] base on adapting 2D movements to the 3D space. These approaches adopt a 3DoF probabilistic motion model in 2D that do not allow accurate modeling of the uncertainty of a 6DoF movement in a 3D space.

One might argue that a motion model might be not be necessary at all. This might be true when the localization algorithm could be executed at such an high frequency that the displacement involved between two subsequent activations of it, is so short that it is reasonable to model the pose uncertainty as normal, and affecting independently the single components of the pose. The larger the displacement between two activations, the larger the uncertainty of the odometric estimate. Such large uncertainty requires, on one hand a very large number of samples (in current state of the art sample-based approaches), on the other - most important - side, it is unrealistically shaped. Conversely, a proper motion model allows to focus samples where it is realistically possible to have the true pose.

In [5] the robot poses are modeled only in 2D, i.e., the state includes only the components x, y, and ϑ (yaw). The other 3 components, i.e., z, the roll angle φ, and the pitch angle ψ, are calculated from the 2D pose estimate and from the structure of the ground surface. Furthermore, the motion does not consider the interactions between the errors acting on the components, and introduce uncertainty on the single components of the movement according to a velocity model. It is to be observed that the independency between the single components of the pose is also assumed in other works [6].

In [4] a representation called multi-level surface maps is used. This technique is proposed as an extension of the elevation maps used in [7] and introduced in [8]. It allows modeling vertical structures within a grid map used for localization with laser range finders. However, these structures do not allow the representation of some typical urban outdoor situations, like bridges or multilevel parkings. Furthermore, in [4] the motion model, more sophisticated then to the one in [5], bases on an evolution of the model introduced in [9], and is similar to that illustrated in [2], again a purely 2D-3DoF motion model.

The inadequacy of these simplifications in urban outdoor situations has driven us to develop a different probabilistic motion model, based on the modeling of a spatial generic movement considering all the components of the 6DoF state. The model, adaptable to different vehicle kinematics, accommodates 6DoF movements even when sensing of some component is missing, e.g., in case of a wheeled vehicle without an IMU.

The next section introduces the proposed motion model, then in section III we compare the our proposal and model that we developed previously, in order to clarify the relevance of an appropriate model. We then conclude presenting experimental data.

II. PROPOSED MOTION MODEL

The model we propose bases on the 2D-3DoF formulation presented in [2, Sect. 5.4]. In that work a displacement is divided in a sequence of 3 steps: an onsite rotation δ_{rot1}, a translation δ_{transl}, and another onsite rotation δ_{rot2}, see Figure 1. This decomposition allows the introduction of the uncertainty on each step in the form of a normal error. These

The authors thank reviewer n. 1 for the comment about the injection of uncertainty in just 3DoF, in the naive motion model; the paper has been changed in this respect.

This work has been supported by the Italian Ministry of University and Research (MIUR) through the PRIN 2009 project *ROAMFREE*

[1]Dept. Informatica, Sist. e Com., Università di Milano - Bicocca, Build. U14 v.le Sarca 336, 20126, Milano, Italy; {galbiati, sacchi}@ira.disco.unimib.it, {ballardini, furlan, sorrenti}@disco.unimib.it

[2]Dept. Elettronica e Informazione, Politecnico di Milano, P.zza Leonardo da Vinci 32, 20132, Milano, Italy; matteucci@elet.polimi.it

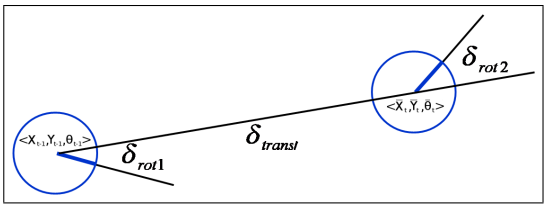


Fig. 1. 2D-3DoF motion model from [2, Sect. 5.4].

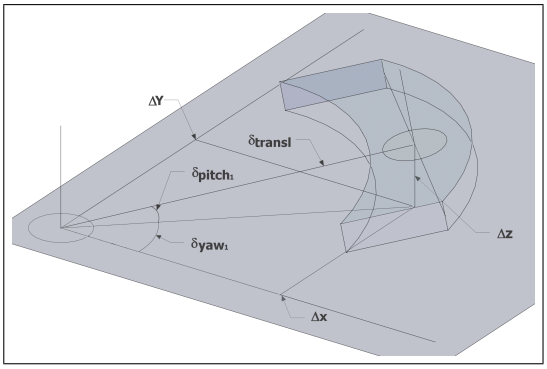


Fig. 2. Uncertainty area of the 3D Model.

errors are zero-mean and are parameterized by a standard deviation that is dimensioned according to the disturbances acting on each step. The composition of the uncertainties of the steps gives a realistic uncertainty affecting the pose, after the application of the motion.

Similarly, the 6DoF motion model has to include motion in all the 6 DoFs of the vehicle and, at the same time, it should divide the whole displacement in a sequence of steps, again zero-mean, so that a parameterized uncertainty on each of the steps turns, after their composition, into a realistic uncertainty on the final pose.

An extended odometer (dead reckoning) can include estimates of displacements in all 6 DoFs, by integrating odometry, i.e., displacements estimated from the rotation of the wheels, with IMU data. In particular, Δx, Δy, and $\Delta\vartheta$ (yaw) can be output by a wheel odometer, while the $\Delta\varphi$ (roll), $\Delta\psi$ (pitch), and Δz, can be output by an IMU. Our proposal includes a set of additional parameters, used when the odometric readings lack some components; e.g., in the case of an IMU-less vehicle.

The state vector has six components, to represent the pose of a rigid body in a 3D world $\mathbf{x}_t = |x, y, z, \varphi, \psi, \vartheta|'$. Let us group the first 3 components in $position_t$ and the last 3 in $orientation_t$, so that $\mathbf{x}_t = |position_t, orientation_t|'$.

Our proposed decomposition of a displacement bases on six steps, which can be grouped in 2 sets: 3 steps to define the new position $\overline{position}_t$ [1], and 3 steps to define $\overline{orientation}_t$. On each step some uncertainty will be added. We now review the first 3 steps, with reference to Figure 2, which give $\overline{position}_t$.

1) Rotation δ_{yaw_1}, which represents a rotation around the Z axis, is necessary to align $orientation_{t-1}$ toward $\overline{position}_t$ in the XY plane; this step corresponds to 2D-3DoF rotation δ_{rot1}.
2) Rotation δ_{pitch_1}, which represents a rotation around the Y axis, and is also necessary to align $orientation_{t-1}$ toward $\overline{position}_t$, but in the XZ plane; this step introduces the possibility of a change in the value of the elevation.
3) Translation δ_{transl}, which represents a translation along the X axis; this translation moves the reference system, after the rotation by δ_{yaw_1} and δ_{pitch_1}, to $\overline{position}_t$; this step corresponds to 2D-3DoF translation δ_{transl}.

The three parameters δ_{yaw_1}, δ_{pitch_1}, and δ_{transl} can be seen as the coordinates, in a spherical coordinate system, of the origin of the new pose $\mathbf{x}_t$. To compute the motion parameters from the extended odometer readings, equations 1 to 3 can be used.

$$\delta_{yaw_1} = arctan(\frac{\Delta y}{\Delta x}) \tag{1}$$

$$\delta_{pitch_1} = arctan(\frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}}) \tag{2}$$

$$\delta_{transl} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{3}$$

For the computation of $\overline{orientation}_t$, our proposal is to compose $orientation_{(t-1)}$ with a generic rotation, which is in turn the composition of 3 last rotation steps. The parameters of these steps, i.e., δ_{roll}, δ_{pitch_2}, δ_{yaw_2}, are sensed directly by the extended odometer.

$$\delta_{roll} = \Delta\varphi; \delta_{pitch_2} = \Delta\psi; \delta_{yaw_2} = \Delta\vartheta \tag{4}$$

In order for the motion model to generate realistic motion uncertainty, it is necessary to add randomness to the components of the state vector $\overline{x}_t$, by acting on the parameters of the motion model. This randomness will be normally distributed, with zero mean. The standard deviation of the components can be calculated according to the following considerations, which are specific to each single step.

1) Rotation δ_{yaw_1}, as in [2], is influenced by:
   - how much the vehicle has rotated, as measured by the wheel odometer;
   - how much space the vehicle has traveled, as measured by the wheel odometer.

   For both factors, the larger the factor, i.e., the change of orientation and/or the traveled distance, the larger the potential mismatch between the odometric measure and real pose.
2) Rotation δ_{pitch_1}, is influenced by:
   - how much the z coordinate has changed, i.e., by Δz, as measured by the extended odometer, from the IMU.
3) Translation δ_{transl} is influenced by:
   - how much space the vehicle has traveled, as measured by the extended odometer; the longer the

[1] With the notation ($\overline{abc}$) we refer to the prediction of the state abc, obtained by the application of the motion model.

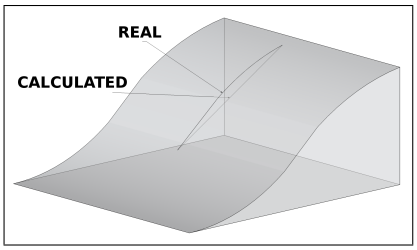


Fig. 3. REAL and CALCULATED (basing on odometry) trajectories, which impact on the uncertainty on δ_{transl}, as due to a change in pitch ($\Delta\psi$).

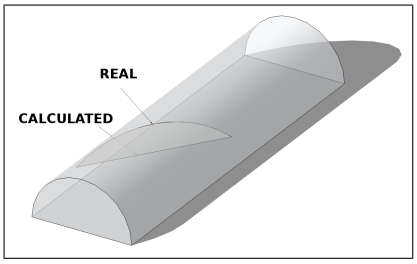


Fig. 4. REAL and CALCULATED (basing on odometry) trajectories, which impact on the uncertainty on δ_{transl}, as due to a change in roll ($\Delta\varphi$).

traveled distance, the larger the potential mismatch between the odometric measure and real pose;

- how much the vehicle has rotated about the Y axis, i.e., the variations $\Delta\psi$, as measured by the extended odometer. A change of pitch while performing a translation, represents a situation where the motion is taking place over a non planar surface. Therefore the traveled distance is larger and the uncertainty is also larger. Figure 3 illustrates the translation resulting from integration of odometry, and the real translation.
- how much the vehicle has rotated about the X axis, i.e., the variation in roll $\Delta\varphi$, as measured by the extended odometer. Figure 4 illustrates the translation resulting from integration of odometry, and the real translation.
- how much the vehicle has rotated about the Z axis, i.e., the variation $\Delta\vartheta$, as measured by the extended odometer. Figure 5 illustrates the translation resulting from integration of odometry, and the real translation.

4) Rotation δ_{roll} is influenced by:
   - how much the vehicle has rotated around its X axis, i.e., variation $\Delta\varphi$, as measured by the extended odometer, from the IMU.
5) Rotation δ_{pitch_2} is influenced by:
   - how much the vehicle has rotated around the Y axis, i.e., the variation $\Delta\psi$, as measured by the extended odometer, from the IMU.
6) Rotation δ_{yaw_2} is influenced by:
   - how much the vehicle has rotated around the Z axis, i.e., the variation $\Delta\vartheta$, as measured by the extended odometer, from the wheel odometer.

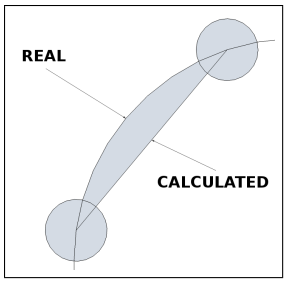


Fig. 5. REAL and CALCULATED (basing on odometry) trajectories, which impacts on the uncertainty on δ_{transl}, as due to a change in yaw ($\Delta\vartheta$).

- how much space the vehicle has traveled: the longer the traveled distance, the larger the potential mismatch between the odometric measure and reality, as measured by the wheel odometer.

Basing on the above mentioned influences, we can define the standard deviations of the noise representing the uncertainty affecting the 6 steps. Finally, in order to gain a better control on the model behavior and similarly to what has been done in [2], we introduce a weight α, for each step.

$$\sigma_{yaw_1} = \alpha_1 \cdot \delta_{yaw_1} + \alpha_2 \cdot \delta_{transl} \tag{5}$$

$$\sigma_{pitch_1} = \alpha_3 \cdot \Delta z \tag{6}$$

$$\sigma_{transl} = \alpha_4 \cdot \delta_{transl} + \alpha_5 \cdot \delta_{yaw_2} + \alpha_6 \cdot (\delta_{roll} + \delta_{pitch_2}) \tag{7}$$

$$\sigma_{roll} = \alpha_7 \cdot \delta_{roll} \tag{8}$$

$$\sigma_{pitch_2} = \alpha_8 \cdot \delta_{pitch_2} \tag{9}$$

$$\sigma_{yaw_2} = \alpha_9 \cdot \delta_{yaw_2} + \alpha_{10} \cdot \delta_{transl} \tag{10}$$

The IMU uncertainty is assumed not correlated with the wheel odometer uncertainty. Moreover, notice that σ_{roll}, σ_{pitch_1}, and σ_{pitch_2} are influenced only by the IMU part of the extended odometer, while σ_{transl} is influenced both by the wheel odometer and the IMU, see Figure 4, 3, and 5.

The sampling motion model will be the following:

$$\hat{\delta}_{yaw_1} = \delta_{yaw_1} + SAMPLE\underbrace{\{\alpha_1 \cdot \delta_{yaw_1} + \alpha_2 \cdot \delta_{transl}\}}_{\sigma_{yaw_1}} \tag{11}$$

$$\hat{\delta}_{pitch_1} = \delta_{pitch_1} + SAMPLE\underbrace{\{\alpha_3 \cdot \Delta z\}}_{\sigma_{pitch_1}} \tag{12}$$

$$\hat{\delta}_{transl} = \delta_{transl} + SAMPLE\left(\underbrace{\begin{array}{l}\alpha_4 \cdot \delta_{transl} + \alpha_5 \cdot \delta_{yaw_2} + \\ \alpha_6 \cdot (\delta_{roll} + \delta_{pitch_2})\end{array}}_{\sigma_{transl}}\right) \tag{13}$$

$$\hat{\delta}_{roll} = \delta_{roll} + SAMPLE\underbrace{(\alpha_7 \cdot \delta_{roll})}_{\sigma_{roll}} \tag{14}$$

$$\hat{\delta}_{pitch_2} = \delta_{pitch_2} + SAMPLE\underbrace{(\alpha_8 \cdot \delta_{pitch_2})}_{\sigma_{pitch_2}} \quad (15)$$

$$\hat{\delta}_{yaw_2} = \delta_{yaw_2} + SAMPLE\underbrace{(\alpha_9 \cdot \delta_{yaw_2} + \alpha_{10} \cdot \delta_{transl})}_{\sigma_{yaw_2}} \quad (16)$$

In case an extended odometer is not available, the expected value will of course be null, and we can use an a priori standard deviation value for each parameter, determined on the basis of the expectations on the change that the terrain can induce in each degree of freedom. Of course this option implies a larger uncertainty, which in turn requires a larger computational effort.

A. Model thresholds

The model exploits a few parameters, i.e., thresholds, in order to handle some situations.

1) Minimum thresholds: As it can be noticed in the relationships above, and similarly to what is done in [2, Sect 5.4], the standard deviations of the uncertainties are proportional to the amount of motion involved into each step. Whenever the motion is too small, the standard deviation gets underestimated. These thresholds are used in such cases; they guarantee a minimum dispersion of the sampled data, which is necessary to correctly represent the real uncertainty.

2) Maximum thresholds: These thresholds have been introduced in order to handle situations where the extended odometer does not give out values in 6DoF, i.e., when there is no IMU on the vehicle. Maximum thresholds represent the maximum a priori uncertainty; on the other hand we expect a better, i.e., more concentrated estimate of robot movements when using a sensor. The σ_{max} value that is associated to every model parameter needs to be suitably large so to ensure that samples can be generated with enough dispersion about the mean value, in order to represent all possible changes on the given degree of freedom. We have chosen the values of these thresholds considering a maximum vehicle speed of 25 Km/h and a 20Hz sampling frequency for the odometer.

III. Comparison with another motion model

In order to clarify the relevance of a careful design of the motion model, we present here also a different model that we developed before the one proposed in this paper. We came first to this model because it was, in our eyes, closely resembling the 2D-3DoF model presented in [2, Sect 5.4]. This model is based on dividing the displacement into 3 steps, see Figure 6.

1) Rotation δ_{rot_1}, a rotation about an axis N_1. To obtain N_1, let us call D the vector $(\overline{position}_t - position_{(t-1)})$. N_1 is the vector product of the X axis of frame $pose_{(t-1)}$ and D.

$$N_1 = (\overline{position}_t - position_{(t-1)}) \times X_{pose_{(t-1)}} \quad (17)$$

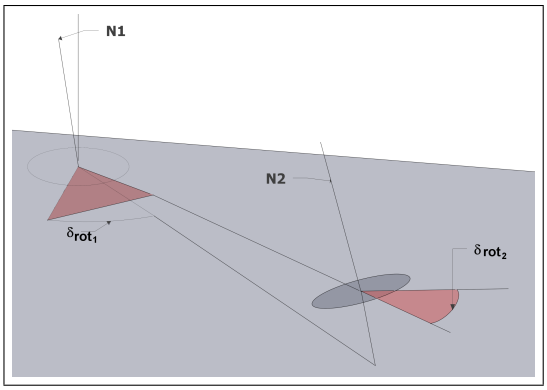


Fig. 6. The first (naive) motion model that was developed.

2) Translation δ_{transl}, a translation along the X axis of the frame obtained at the previous step, i.e., after the rotation of $pose_{t-1}$ by δ_{rot_1}. At the end the origin will reach $\overline{position}_t$.
3) Rotation δ_{rot_2}, a rotation about an axis N_2. N_2 is the vector product of the X axis of frame $pose_t$ and D.

$$N_2 = (\overline{position}_t - position_{(t-1)}) \times X_{pose_t} \quad (18)$$

This rotation aligns the reference frame, which has been obtained rotating $pose_{(t-1)}$ by δ_{rot_1} and then translating by δ_{transl}, to $\overline{orientation}_t$.

The uncertainty on the components of the motion model is sampled from normal distributions, for each of the 3 parameters δ_{rot_1}, δ_{transl}, and δ_{rot_2}. Such distributions have zero-mean and standard deviations computed similarly to what has been done for the motion model in [2, Sect 5.4]. It is just a similarity because of the need to introduce other degrees of freedom to the uncertainty affecting $\overline{position}_t$, which would be just 2 (δ_{rot_1}, and δ_{transl}), for a 3D point. We therefore add noise to the vector N_1 as, if we added noise to the vector D, we would obtain the model proposed above. Notice that the 2 parameters of N_1 are not independent w.r.t. the uncertainty of rotating about N_1, so the DoF count for $\overline{position}_t$ is correct.

This naive model, which turned out not being well performing, demonstrates how heavily the decomposition of the overall displacement, i.e., the motion model, affects the capability to produce realistic poses. Actually, the poses generated by this model are not realistically distributed about the real pose, see Figure 7 and Figure 8, where it can be observed that the uncertainty is rotated along the X axis; the larger Δz, the more rotated the particle cloud. Figure 9 shows the corresponding uncertainty for the motion model proposed above.

IV. Experimental Results

We first tested the software implementation of the proposed motion model in simulation; because of space limits we report hereafter only some real tests performed on our research vehicle. Testing have been performed is the parking area of the U5 building of Università di Milano - Bicocca, see Figures 10, 12, 13.

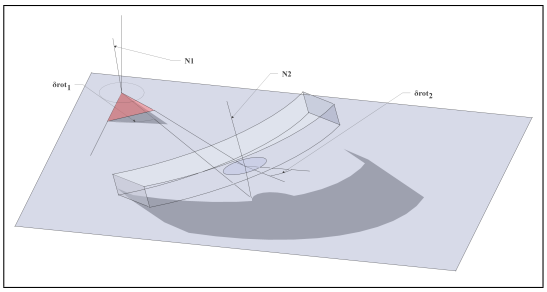


Fig. 7. Uncertainty area for the naive motion model.

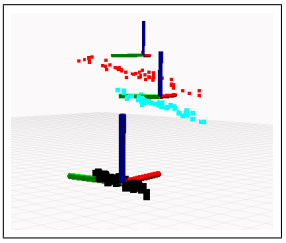


Fig. 8. 3D view of the particle set for the naive motion model. Notice that the larger the overall Δ_{yaw} and Δz, the larger the distortion of the particle set.

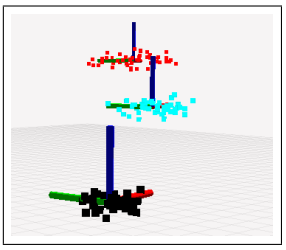


Fig. 9. 3D view of the particle set for the proposed motion model. Notice the absence of the distortion that is affecting the particle set presented in Figure 8

The motion model have been plugged into a state of the art Monte Carlo Localization software [10]. We performed the tests in this order: first we verified that the localization was correct when moving on an almost planar surface, i.e., the model was performing at least as the state of the art 2D-3DoF model. This has been done in the underground garage of the building, where the floor appears to be reasonably planar. We did obtain results comparable to the ones obtained with the 2D-3DoF state of the art software [10]. Secondly, we drove along a path including a ramp, from the garage to the outdoor parking area, see Figure 11. Also in these last tests the localization was successful, one experiment is depicted in Figure 14. As we have no ground truth for the pose, we checked that at the end of the path, at about pose n. 8 in Figure 10, the estimated pose was matching the real one, and we always obtained this result.

On the other hand, the naive motion model fails in high curvature curves, as it might be expected from observing, in Figure 7, the unrealistic uncertainty implied by this model; an example of failure is presented in Figure 15.

Despite roll and pitch data were available, thanks to an MTi X-sens IMU sensor, we verified that using only the available LIDAR sensors, altogether with appropriate

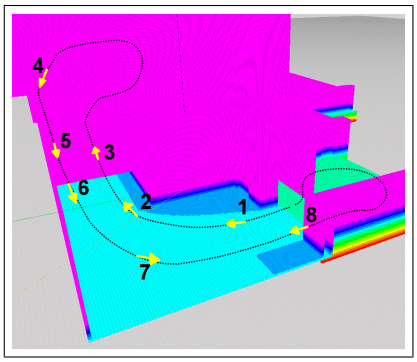


Fig. 10. Voxel representation of the U5 building underground garage. Here it is shown the part that includes the ramp leading to the outdoor parking: pose n. 8 is at the gate of the underground garage, pose n. 4 is in the outdoor parking nearby where the cart is depicted in Figure 13, poses n. 3 and 5 are on the ramp, poses n. 1, 2, 6, 7 are in the road leading to the ramp.

minimum and maximum thresholds, sufficed for a correct localization. We also noticed that using together both types of LIDAR models we had available (Sick LMS111 and LDMRS4001) is extremely useful, since they measure on different scanning planes.

CONCLUSIONS

We presented a motion model for 3D-6DoF localization, and showed that a careful design is required to obtain a realistic representation of the involved uncertainties. The presented model demonstrated its suitability in different experiments and is currently in use for our research in urban autonomous driving.

REFERENCES

[1] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.
[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.
[3] C. Olson, "Probabilistic self-localization for mobile robots," *Robotics and Automation, IEEE Trans. on*, Feb 2000.
[4] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte carlo localization in outdoor terrains using multi-level surface maps," in *In Proc. of the Int. Conf. on Field and Service Robotics*, 2007.
[5] T. Suzuki, M. Kitamura, Y. Amano, and T. Hashizume, "6-dof localization for a mobile robot using outdoor 3d voxel maps," in *IROS*, 2010.
[6] I. Baldwin and P. Newman, "Road vehicle localization with 2d push-broom lidar and 3d priors," in *ICRA*, Minnesota, USA, May 2012.
[7] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous rover navigation on unknown terrains: Functions and integration," *Int. Jour. Robotics Research*, vol. 21, no. 10-11, 2002.
[8] A. I. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *21st Int. Conf. Machine Learning*, 2004.
[9] N. Roy and S. Thrun, "Online self-calibration for mobile robots," in *ICRA*, 1999.
[10] B. Gerkey, "Amcl." [Online]. Available: http://ros.org/wiki/amcl

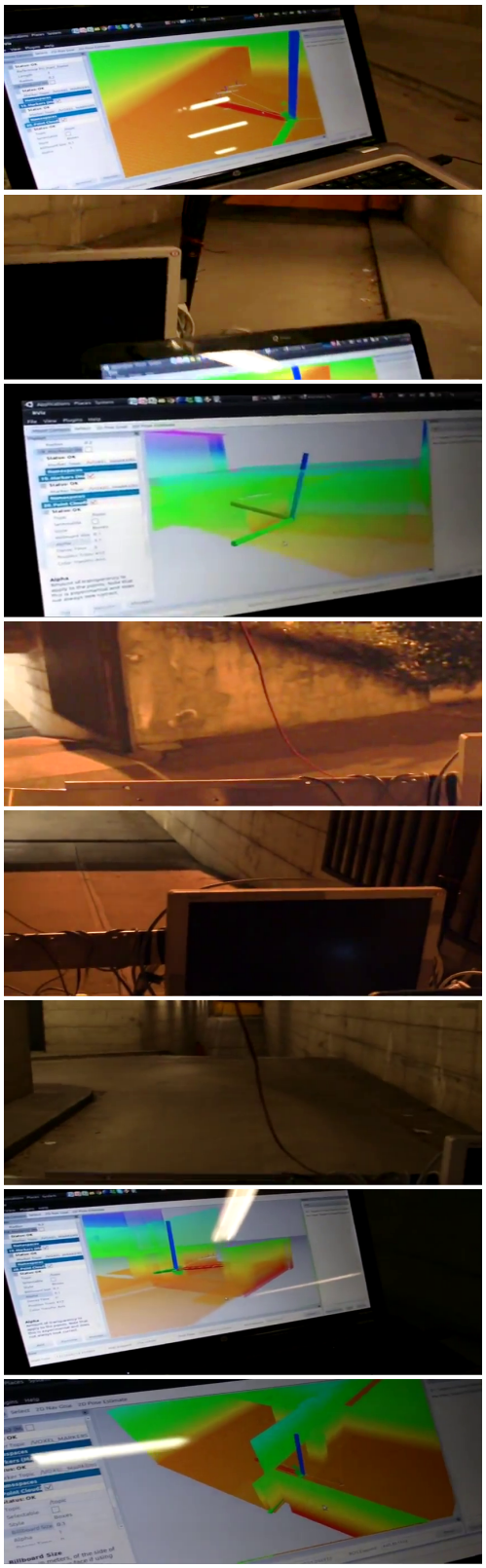


Fig. 11. Snapshots along the path shown in Figure 10: from the underground garage, through the ramp, to the outdoor parking lot and back into the garage; 1) top to 8) bottom. Notice in 1), 3), 7) and 8) the cart reference frame, shown in the same map used by the software. Notice in 3) the frame pitching up along the ramp.

Fig. 12. U5 building - Aerial view, the ramp from the underground garage can be noticed on the left of the largest tree.

Fig. 13. The ramp from the underground garage to the outdoor park, picture taken from the outdoor park.

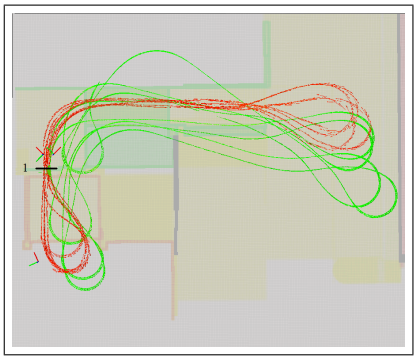


Fig. 14. The green path represents the odometric path; the red path represents the localization obtained using the proposed motion model. Notice at 1) i.e., nearby pose n. 8, the correctness of the localization.

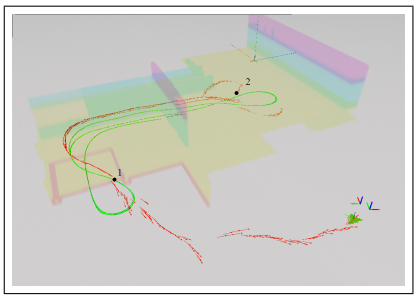


Fig. 15. A typical experiment with the naive motion model. In the path, just before 2), the localization system failed (whether this error is tolerable is not analyzed here), but in 2) it could luckily recover; in 1) it fails completely, without recovery.

Visual trajectory learning and following in unknown routes for autonomous navigation

David A. Márquez-Gámez and Michel Devy

Abstract— **This paper describes the design and testing of a system to enable large scale cooperative navigation of autonomous vehicles moving on a priori unknown routes. A large-scale learning-mapping approach and a map-based replay-localization method are combined to achieve cooperative navigation. The learning approach is based on a proposed hierarchical/hybrid BiCam SLAM and the replay approach exploits a localization method based on an active search procedure. The system can be generalized to be executed on multiple vehicles moving as a convoy. A global $3D$ map maintains the relationships between a series of local maps built by the first vehicle of the convoy (leader), defining a path that all other vehicles (followers) must stay on. Only single camera setups are considered. The overall approach is evaluated with real data acquired in an urban environment.**

I. INTRODUCTION

Many robotic missions can be more efficiently and robustly achieved by a team of robots. This paper is focused on the cooperation of several vehicles, moving alone or as a convoy in open environments, on a priori unknown routes. Although most existing mobile robotic applications involve a single robot, a wide range of potential applications require multiple robots to execute joint tasks, e.g. rescue robotics, cooperative monitoring [1], [2], [3]

Localization is a key technology to address how the robots localize themselves in the operating unknown environment and how they know their local poses with respect to other objects. A variety of approaches have been reported for localization of multirobot formations. In [4], the localization problem of a leader-follower system, is based on EKF to estimate each follower's pose with respect to the leader. In [5], it is proposed a behavior-based approach for maintaining formation of a team of robots, with experiments performed on outdoor unmanned ground vehicles equipped with vision, GPS and hazard sensors. In [6], a scheme for distributed outdoor localization for a team of robots is based on heterogeneous sensors such as local area differential global positioning system (LADGPS) and cameras. Most of the above approaches focused on the relative localization only, and a few of them discussed how to globally localize robots. Many methods simply assume that robots are equipped with absolute positioning capabilities.

We present a complete system for large scale autonomous operation of a team of robots in a "convoy" formation navigating in a dangerous, unknown and changing outdoor environment, typically on routes that could be mined. The proposed system consists of 2 steps: (1) learn first a safe path using an unmanned robot, and (2) follow this same path by other vehicles of the fleet. The two steps involve different functions: the initial path learning requires SLAM, i.e. to simultaneously memorize successive robot positions on the path and landmark positions that are used by the robot to locate itself. When the leader's path is traveled again by other vehicles of the convoy, the map will be exploited to localize and control a vehicle so that it is maintained on the same path, with a tolerance that must be minimized.

It is assumed that all vehicles are only equipped by cameras and odometers, used during the learning or replay steps. The temporal gap between the two steps depends on the mission. (1) the learning and replay functions could be executed independently, i.e. the map and the path are acquired at time A, and the safe trajectory is followed again later at time B, with $B-A$ equal to several hours or days. In such a situation, the environment could change between the two steps: recorded landmarks could be removed during the interval, or the path itself could get blocked. Moreover vehicles used at time A and B could be the same, or could be different, involving the use of different cameras, so possibly different radiometric information. (2), these functions could be executed on different communicating vehicles navigating in a convoy formation. A leader vehicle records the map and the path, and sends these information to the second vehicle of the convoy using a wireless network; the first follower stays on the path taken by the leader, updating the map from its own observations and sends the updated map and the path to the next follower. Mutual localization (the follower sees the leader) is possible, but not mandatory. The leader and followers are equipped with different cameras, involving possible radiometric differences on images.

The contribution of this paper concern large scale navigation in a "convoy" formation in unstructured, three-dimensional terrain. Only perception is considered. Moreover, the environment is assumed to be static (no dynamic obstacle), but it can be modified between the learning and replay steps.

In section II the system description is presented. In section III the learning-mapping approach is described. In section IV the replay-localization method is presented. Section V gives experimental results for the proposed system using a real robot. Finally in section VI, current results and conclusions are discussed.

II. SYSTEM DESCRIPTION

The major processing blocks of our system are depicted in Fig. 1. The system consists of 2 steps: (1) the learning-mapping step, where a leader vehicle, builds several $3D$-points maps and learn a safe path, and (2) the replay-localization step, where the information learned and trans-

CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France. Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; Toulouse, France {dmarquez, michel} at laas.fr

mitted by the leader, allows other vehicles of the fleet, to follow the learned safe path. The two steps involve different functions: the initial path learning is based on a proposed hierarchical SLAM using only vision in a BiCam configuration and the replay procedure exploits a localization method based on an active search procedure to localize and control a vehicle so that it is maintained on the same path than during the learning step, with a tolerance that must be minimized. In sections III and IV learning and replay procedures are described.

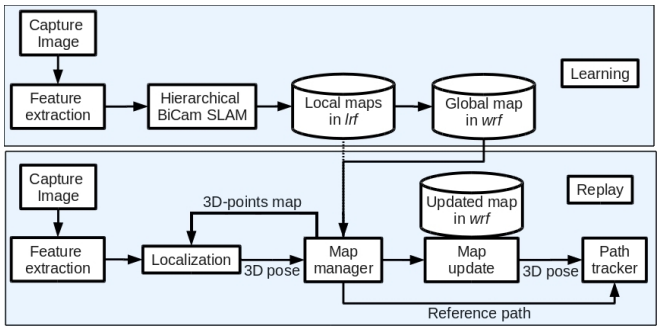


Fig. 1. An overview of the major processing blocks in our proposed system

III. Learning step: learn a safe trajectory

A. Hierarchical BiCam SLAM

Our application involves motions of robots in large environments: so the mapping problem is formulated using submaps in a similar manner to hierarchical SLAM [7]. Independent consecutive local maps are represented in their own reference frame *(lrf)*, and the upper global level *(wrf)* is a graph whose nodes correspond to local maps, and whose edges are annotated by relative transformations between *lrf*s. The principle is represented on Fig. 2.

1) Local Level: The local level contains the locally referred stochastic maps of landmarks, built with the BiCam SLAM algorithm [8]. Here a landmark is a 3*D* point, observed as an interest point in images. Each point is linked to an image patch which is defined as a small image region (traditionally 11*x*11 pixels). The *k*-th local map is defined by $\mathbf{x}_k^L = [\mathbf{x}_k, \mathbf{g}_k]^\top$ (superscript "L" stands for local map) where $\mathbf{x}_k$ is the current pose of the robot, and $\mathbf{g}_k = [\mathbf{l}_1^k \cdots \mathbf{l}_m^k]^\top$ is the set of m mapped landmarks, both with respect to the *k*-th *lrf*. The camera and robot positions are linked by a known rigid transformation, so that BiCam SLAM can keep a Gaussian estimate $\mathbf{x}_k^L \sim \mathcal{N}\{\hat{\mathbf{x}}_k^L, \mathbf{P}_k^L\}$ of this map, namely

$$\hat{\mathbf{x}}_k^L = \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{g}}_k \end{bmatrix}, \quad \mathbf{P}_k^L = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k\mathbf{x}_k} & \mathbf{P}_{\mathbf{x}_k\mathbf{g}_k} \\ (\mathbf{P}_{\mathbf{x}_k\mathbf{g}_k})^\top & \mathbf{P}_{\mathbf{g}_k\mathbf{g}_k} \end{bmatrix} \qquad (1)$$

Maps are built sequentially. Once a threshold is reached, either in number of landmarks or in robot uncertainty, the current map $\mathbf{x}_k^L$ is closed and a new map $\mathbf{x}_{k+1}^L$ is created, starting in a new *lrf* with robot pose $\hat{\mathbf{x}}_{k+1}$ and error covariance equal to zero. Each landmark known in $\mathbf{x}_k^L$ reobserved when the new map $\mathbf{x}_{k+1}^L$ is created, keeps the same label.

2) Global Level: The global level is represented as an adjacency graph in which local maps $\mathbf{x}_k^L$ in *wrf* are nodes, and the edges between them are annotated first by the relative transformations between successive *lrf*s, noted $\mathbf{w}_k^{k+1}$, and secondly, by the list of labels of common landmarks, between $\mathbf{x}_k^L$ and $\mathbf{x}_{k+1}^L$. Let us define the global level as the Gaussian state $\mathbf{w} \sim \mathcal{N}\{\hat{\mathbf{w}}; \mathbf{P}_w\}$ of relative transformations between local maps, namely:

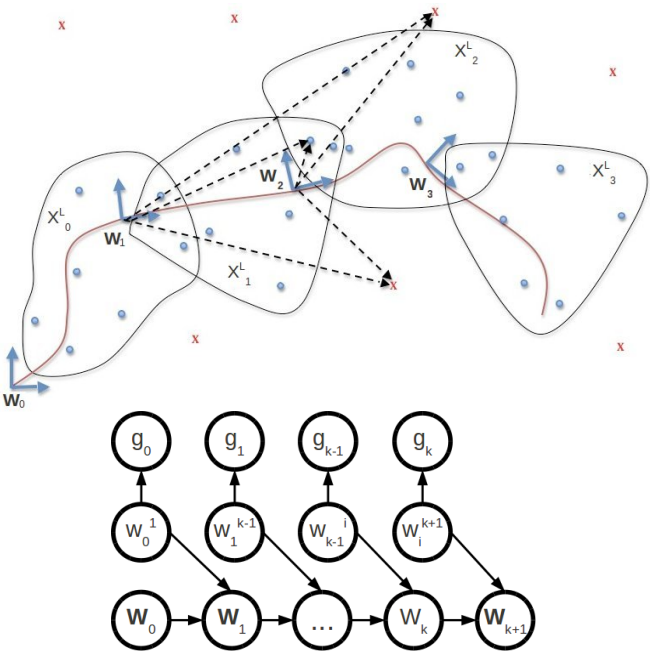


Fig. 2. The map: a set of submaps with their own *lrf*s, and with common landmarks, known either by euclidean coordinates for close points (blue dots) or by IDP vectors generally for points at infinity (red X). Graphical model as a simplified Bayesian network: submaps constitute the local level $(\mathbf{x}_k, \mathbf{g}_k)$ and the global level on top links the local submaps $(\mathbf{W}_k)$. The map building is sequential, corresponding to the robot exploring and not closing loops, having always $\mathbf{w}_{k+1}^k = \mathbf{x}_k$.

$$\hat{\mathbf{w}} = \begin{bmatrix} \hat{\mathbf{w}}_0^1 \\ \vdots \\ \hat{\mathbf{w}}_k^{k+1} \end{bmatrix}, \quad \mathbf{P}_w = \begin{bmatrix} \mathbf{P}_{w_0^1} & 0 & 0 \\ 0 & \cdot & 0 \\ 0 & 0 & \mathbf{P}_{w_k^{k+1}} \end{bmatrix} \qquad (2)$$

Let us note $\mathbf{W}_k$ the origin of the local map $\mathbf{x}_k^L$, expressed in the *wrf*; it can be computed by compounding relative transformations from $\mathbf{w}_0^1$ to $\mathbf{w}_{k-1}^k$. The global level can be viewed as a sparse pose-SLAM as in [9], where local maps are like landmarks hanging from robot poses in *wrf*. The global state in our case contains transformations $\mathbf{w}_k^{k+1}$, instead of absolute poses $\mathbf{W}_k$ (Fig. 2).

B. Off line refinement of a map to be used later

In the case of a path recorded in order to be run later, the final model is memorized as a hierarchical map. For an edge between nodes i and $i+1$, a common landmark l has two representations $l^{(i)}$ in the *lrf* i, and $l^{(i+1)}$ in the *lrf* $i+1$. This model must be refined or transformed before to be exploited later for navigation.

1) Refinement of the global map: In our context, a path follows a route between two different areas, so that a classical Loop Closure is not possible; the global graph has no cycle. Nevertheless, the global map can be refined, by non linear optimization. Successive submaps can be made consistent, computing optimal and unic representations for all common landmarks and optimizing transforms between submaps.

We formulate the refinement of the global map as a local submap joining problem in a similar manner that in [10]. The algorithm is generalized and formulated as a least squares optimization problem and solved by Extended Information Filter (EIF) together with smoothing and iterations.

A local map is defined by equation (1). Since the local map provides a consistent estimate of the relative position from robot poses to local features, this map can be treated as an observation made from the robot start pose to all the features in the local map and a virtual robot located at the robot end pose. The observation value is the local map state estimate and the observation noise is a zero-mean Gaussian *pdf* with the covariance matrix equal to the local map covariance matrix. Thus, the refinement of the global map becomes a large-scale estimation problem with only local maps information.

The Extended Information Filter (EIF) is used to solve the estimation problem. A non zero off-diagonal element in the information matrix (a link between the two related objects) occurs only when the two objects are within the same local map. Since the size of each local map is limited, any object will only have links with its nearby objects no matter how many (overlapping) local maps are fused (Fig. 2). This results in an exactly sparse information matrix similar to Smoothing and Mapping (SAM) [11].

IV. Replay step: Replaying the learnt trajectory

Using the map database (a set of optimized local maps) produced by the leader robot, the followers robots are able to repeat a learned path any number of times. Two modes are considered: (1) in the replay mode the trajectory is globally stored in a database by the leader, and executed again long time after by another vehicle; (2) in the convoy mode, submaps are successively transmitted from the leader to the followers. Each follower reexecutes the path attached to each submap, and possibly refines the landmark positions, overall if it can observe the previous vehicle of the convoy. During the execution, every $3D$-point landmark of the map is sent to an *active search* procedure. If a landmark is not matched a number of times, then, it is removed from the map and replaced by another one initialized in the same image region.

A. Localization from the learned map

Before a vehicle replays a learnt path, it must be able to localize itself from an initial position close to that the leader initially had. Indeed, it is assumed that the follower vehicle starts from a position close to that of the leader but not necessarily identical. Then the follower must compare the map information with its own perception of the environment and compute its localization.

1) Initial position: In our approach, the follower vehicle has to localize itself and navigate thanks to a $3D$ map with accuracy and in real time. The initialization step is a critical point because the vehicle location is not accurately known in the map. In order to focus on its real localization, it perceives its environment through its camera. So, each image patch associated to the $3D$ point supposed to be observed is actively searched within an elliptical region. The latter is based on the projection of the uncertainty of the considered point taking into account the vehicle position. A cross-correlation measure based on ZNCC (Zero Normalized Cross Correlation) is used ([12]). While this search is performed, the best score for each image patch is stored, which provides a first hypothesis of matching. For each pair of $3D-2D$ points, an update step is calculated through a Kalman filter (details are explained in next section). Once the update, the difference between the estimated projection of each point of the map in the image and the result of the cross-correlation measure decreases. This allow the selection of a first $2D-3D$ couple.

2) On-line Localization: At this stage a good estimate of the starting point is available. Thanks to the fairly close estimate of the operating point, we can rely on a Kalman filter to estimate the position with proprioceptive sensors and refine it by using observations collected during the motion. This method is fully described and evaluated in [13].

2.1) Prediction step: The first step of the Kalman filter consists in the prediction through proprioceptive data. It establishes a model of evolution of the vehicle using a tricycle model. The equation model is:

$$\begin{cases} X_{k+1} &= X_k + ds\cos(\theta_k+\alpha/2) &+ \varepsilon_X \\ Y_{k+1} &= Y_k + ds\sin(\theta_k+\alpha/2) &+ \varepsilon_Y \\ Z_{k+1} &= Z_k + ds\sin(\phi) &+ \varepsilon_Z \\ \theta_{k+1} &= \theta_k &+ \varepsilon_\alpha \\ \phi_{k+1} &= \theta_k &+ \varepsilon_\beta \\ \psi_{k+1} &= \theta_k &+ \varepsilon_\gamma \end{cases} \quad (3)$$

where θ, ϕ, ψ are respectively the *yaw*, *pitch* and *roll* angles. The length ds is a function of a priori traveled distance and parameters which are specific to the vehicle. It is supplied by odometry measurements. The angle α reflects the constant steering angle between times k and $k+1$ and the parameters of the vehicle. The uncertainty in this estimate is provided by the Jacobians of the evolution model. The variance-covariance matrix is then expressed as, $\mathbf{P}_{k+1|k} = \mathbf{F}_X \mathbf{P}_{k|k} \mathbf{F}_X^\top + \mathbf{Q}_{k+1}$, where $\mathbf{Q}$ is the covariance of process noise and $\mathbf{F}_X$ is the Jacobian of the evolution model, with respect to X.

2.2) Data association step: The search for correspondence is limited by the projection of the uncertainty associated with the movement of the vehicle and the uncertainty of the position of the $3D$ point. A RANSAC algorithm, which is similar in part to that described in [14], can reject absurd tracking results and find the correct matchings.

2.3) Estimation step: This prediction is then refined with the 2D observations of the $3D$-world points. The association provides information on data matching between the points $(u_{obs}, v_{obs})^T$ of the 2D image and $3D$ points P_{3D} of the map. Let us define the rigid transformation between the world and the reference frame of the vehicle as: $\mathbf{R} = R_z(\theta_k)R_y(\phi_k)R_x(\psi_k)$ and $T = (X_k, Y_k, Z_k)^\top$. The $3D$ points of the map are projected in the image with the linear relationship, $P_{2D} = \mathbf{K}\mathbf{R}^\top(P_{3D} - T) = (ku, kv, k)^\top$, where $\mathbf{K}$ is the intrinsic parameter matrix of the camera.

The estimated coordinates are divided by the scale factor, $u_{est} = ku/k$ and $v_{est} = kv/k$

$$u_{est} = h_u(P_{3D}) = \frac{K_1 \mathbf{R}^\top (P_{3D} - T)}{K_3 \mathbf{R}^\top (P_{3D} - T)}$$
$$v_{est} = h_v(P_{3D}) = \frac{K_2 \mathbf{R}^\top (P_{3D} - T)}{K_3 \mathbf{R}^\top (P_{3D} - T)} \quad (4)$$

where $\mathbf{K}_i$ is the i-th line from the intrinsic parameter matrix and h_u, h_v are the part of the observation function related to the u and v coordinates.

The covariance of the innovation of the Kalman filter ($\mathbf{H}_X$), is obtained using the Jacobians of the observation model. The Jacobians are calculated, using equation (4). The Kalman gain associated with a pair can be calculated by, $\mathbf{G}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{H}_X{}^\top (\mathbf{H}_X \mathbf{P}_{k+1|k}\mathbf{H}_X{}^\top + \mathbf{R}_{obs})^{-1}$, where $\mathbf{R}_{obs}$ denotes the covariance of the noise associated with the observation, in pixels.

Finally, the data is updated from observation by:

$$X_{k+1|k+1} = X_{k+1|k} + \mathbf{G}_{k+1}\left(\begin{pmatrix} u_{obs} \\ v_{obs} \end{pmatrix} - \begin{pmatrix} u_{est} \\ v_{est} \end{pmatrix}\right) \quad (5)$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{G}_{k+1}\mathbf{H}_X\mathbf{P}_{k+1|k} \quad (6)$$

This step is for each pair of $3D-2D$ points. Localization accuracy depends on the number of points, the accuracy of the initial positioning and of the learnt map.

B. Path tracker: trajectory following

Path tracker module have been developed in a similar manner to *Tiji* [15], to integrate the replay step algorithm into our proposed system. Fig. 3, shows the path tracker processing block.

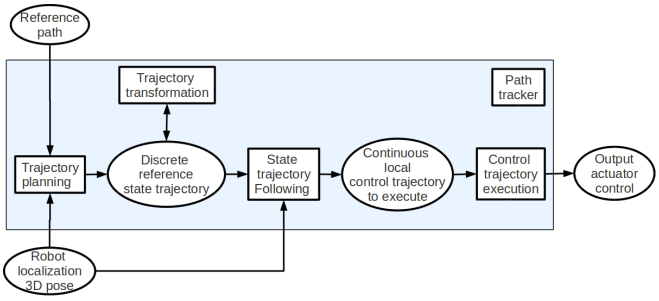


Fig. 3. An overview of the path tracker processing block in our proposed system

Path tracker is an algorithm that computes a feasible trajectory between a start and a goal state. The method proposed, which relies upon a parametric trajectory representation, is variational in nature. The trajectory parameters are incrementally updated in order to optimize a cost function involving the distance between the end of the trajectory computed and the desired goal. Should the goal state be unreachable (eg. if the final time is ill-chosen), path tracker returns a trajectory that ends as close as possible to the desired goal.

1) Trajectory planning: This module provides the algorithm to plan and apply geometric transformation over a discrete trajectory. This module does not include any global motion planning approach as the current experiments have been focused on repeating a learnt trajectory.

2) Trajectory following: This module encapsulates the navigation algorithm. Taking as input the robot localization provided by the replay-localization and the discrete representation of the reference path provided by the learning step, this module provides a reactive method trying to follow a reference trajectory while providing at each time step a parametric continuous control trajectory (continuous sequence of input commands) until a given time horizon.

3) Control trajectory: Given the continuous control trajectory provided by the previous step, the *Control trajectory* module is aimed at checking if the control is still valid and admissible for the robotic system (respects its motion constraints) and computing a command to provide as input of the actuators.

V. Experimental Results

A. Learning and Replay

For this experiment, we have tested our system in the parking lot of the LAAS-CNRS (see Fig. 4) using a ground robot (Segway platform) equipped with two independent stereo-vision benches. The first one used in the *Learning* step is made of two Marlin 1280 x 960 cameras. The second one used in the *Replay* step is made of two Flea2 1280 x 960 cameras (see Fig. 5). So the experiment was undertaken with the same robot but with different cameras between the learning (with Marlin cameras) and replay (with Flea2 cameras) steps, exploited in a BiCam configuration. Moreover the experiment was undertaken with path learning and replay, done on different time, thus, the environment has changed between the two steps.

The robot follows the trajectory shown in Fig. 4. The main issues of this experiment are, first, learn a safe path, computing the learning-mapping approach and second, replay the path autonomously using the replay-localization and the path tracker method. During the first step, the learning-mapping procedure was used to map $3D$-points landmarks and build the submaps. New local maps are created when 80 landmarks are in the map. In each submap, the poses and their uncertainties are expressed in the associated *lrf*. All submaps were merged in a final global map, expressed in *wrf* using the refinement of the global map procedure (see Fig. 6). As a second step to test the replay-localization approach, the replay-robot must replay the path learned during the learning-mapping procedure. We can note that the initial position of the robot in the replay step is not exactly the same that the initial position in the learning step. The $3D$ global map and the global path are loaded on the robot. In this point, the replay-robot, has a good knowledge of the $3D$ map of the environment, thus, the robot starts with a huge uncertainty and applies the algorithm of replay-localization. The results obtained in this experiment[1], confirm the system performance in the learning-mapping procedure as well as the accuracy of the replay-localization along the path (Fig. 7). We can see that the replay step works in spite of bad matchings on landmarks occluded or lost, due to the environments changes (Fig. 8).

[1]A representative video of the experimental results can be seen at `http://homepages.laas.fr/dmarquez/maplaas`

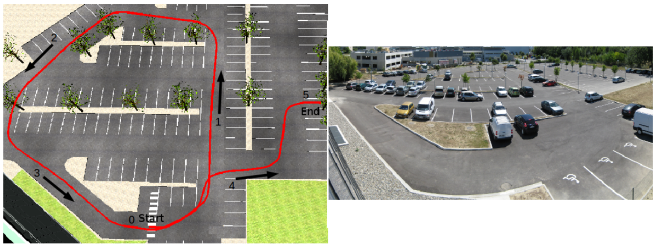


Fig. 4. View of the learned and replayed trajectory in the experiment. The robot follows the trajectory described by the sequence 0-1-2-3-4-5.

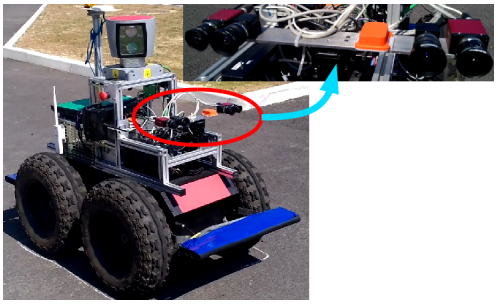


Fig. 5. The Segway platform used in the experiment. Two independent stereo-vision benches. The first one made of two Marlin cameras (red cameras) with a baseline of 0.40 *m*. The second one made of two Flea2 cameras (black cameras) with a baseline of 0.30 *m*.

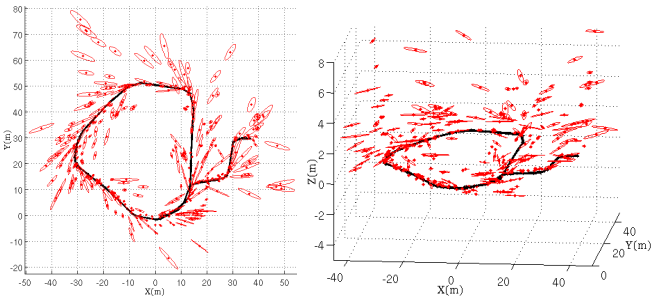


Fig. 6. Learning and Replay experiment: 2*D* and 3*D* plots of the global map obtained by joining 67 local submaps using our hierarchical/hybrid BiCam SLAM approach with *refinement of the global map* procedure; red: covariance ellipses of the features; black: the estimated robot trajectory.

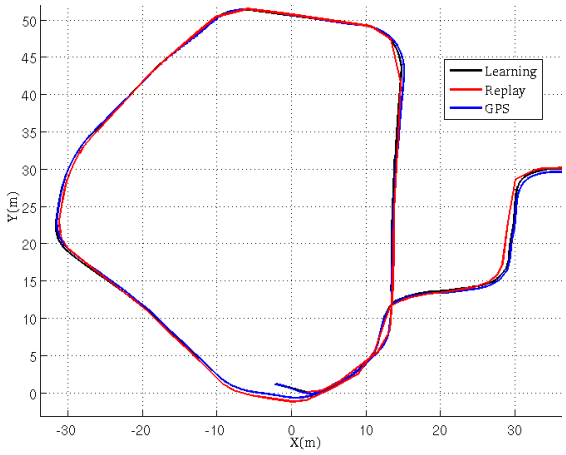


Fig. 7. Learning and Replay experiment: robot trajectory estimate; black: the robot trajectory learned path; red: the robot trajectory replayed path; blue: the true robot trajectory (GPS).

B. The Convoy task

A second experiment using MORSE[2] was undertaken. MORSE is a domain independent simulator, where virtual

Fig. 8. Learning and Replay experiment: images acquired on the same area. (left) learning-mapping step; *magenta*: IDP landmarks; *dark red*: IDP landmarks not observed in the frame; *cyan*: euclidean landmarks; *dark blue*: euclidean landmarks not observed in the frame. (right) replay-localization step; *magenta*: initial projections of 3*D* points from the learned 3*D* map; *cyan*: projected and matched points using for localization.

robots can interact with a 3*D* environment, using sensors and actuators that behave in the same way as their counterparts in the real world. The main issues of this experiment is to make a "Convoy" with two robots (leader and follower) where the leader applies the learning-mapping approach to build successive submaps, and to transmit online each submap once it is built, defining a path that the follower must stay on. The follower, waits for the first submap to start and applies the replay-localization approach to follow the trajectory, processing the information received (online) from the leader. Thus the leader-robot start to explore the area and build the first local map, this first submap is the origin of the world. New local maps are created when 80 landmarks are in the map, but before starting a new local map, the submap and the local trajectory are expressed in *wrf* and transmitted to the follower-robot. Immediately after, the follower-robot pose is the new relative transformation in the global graph. As results, 20 submaps were generated and sent from the leader to the follower. The follower was able to process each submap and apply the replay-localization approach to track and follow (online) the path defined by the leader (see Fig. 9). The results obtained in this experiment, confirm the system performance in the convoy task (see Fig. 10)

VI. Conclusions

We have designed and tested a navigation system in order to implement convoy navigation by robots that must navigate in dangerous unknown routes. Only perception results have been exhibited, a general method for trajectory control has

[2] http://morse.openrobots.org

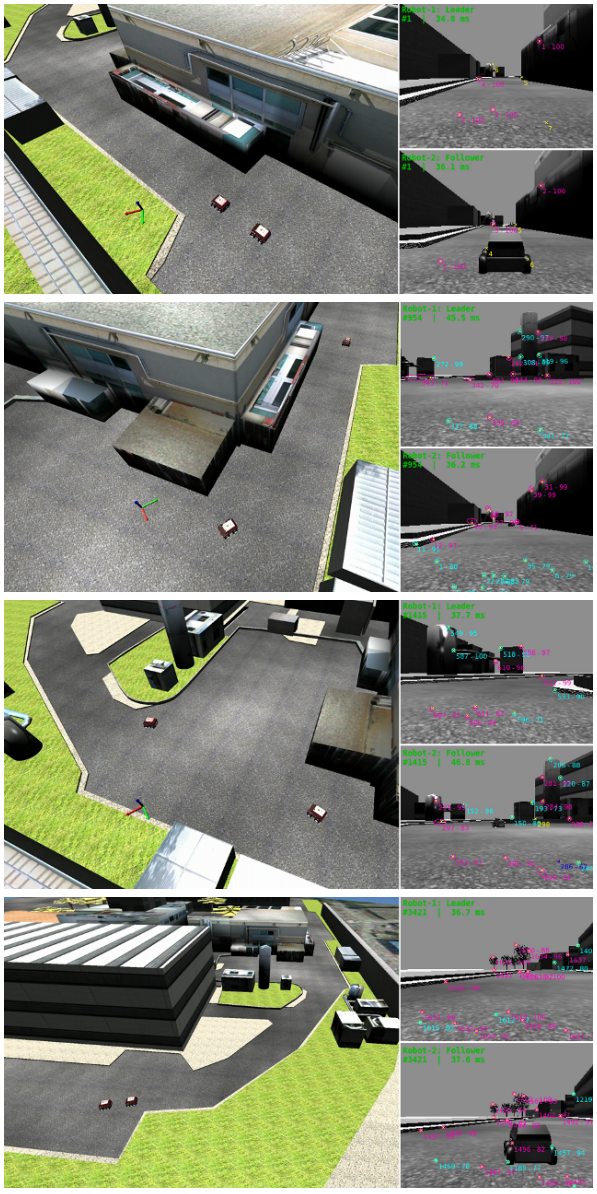


Fig. 9. Convoy experiment: (left) global view of the experiment scenario; (right-top) leader view, learning-mapping step; *magenta*: IDP landmarks; *cyan*: euclidean landmarks. (right-down) follower view, replay-localization step; *magenta*: initial projections of 3*D* points from the learned 3*D* map; *cyan*: projected and matched points using for localization.

been used to maintain the follower on the learnt trajectory. A mixed strategy combining path tracking with visual servoing will be studied in the future.

It has been shown [16] that visual SLAM methods based on non linear optimization (Incremental Sparse Bundle Adjustment, or SBA) converge better than EKF-SLAM or more generally, methods based on filtering. Here why do we use an EKF-SLAM method? First in this application, loop closure is not considered: the method must both build the map and the trajectory, but the evaluation criterion will be based on the capability for a robot to replay a learned trajectory, whatever the drift with respect to the exact localization in a global frame. Non linear optimization is only used off line in order to refine the map and the learned trajectory. Then for the

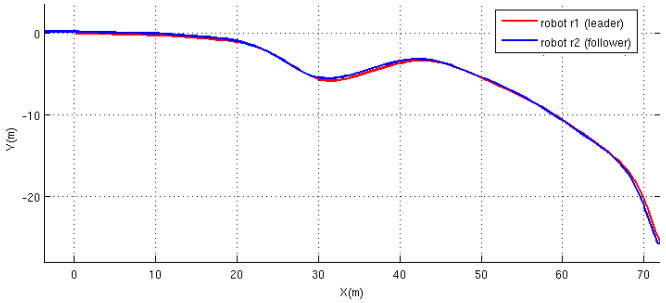


Fig. 10. Convoy experiment: Robot trajectory estimate; blue: the robot trajectory learned path; red: the robot trajectory replayed path.

convoy configuration, off line refinement is forbidden. Thus, in our system, EKF-SLAM has been selected has the best way to generate submap's that could be communicated from the leader robot to the followers ones.

References

[1] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 2, pp. 223 – 237, Apr. 2003.

[2] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 2, pp. 163 –175, 2009.

[3] L. Vig and J. Adams, "Multi-robot coalition formation," *Robotics, IEEE Transactions on*, vol. 22, no. 4, pp. 637 –649, 2006.

[4] G. Mariottini, G. Pappas, D. Prattichizzo, and K. Daniilidis, "Vision-based localization of leader-follower formations," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 635 – 640.

[5] T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 6, pp. 926 –939, Dec. 1998.

[6] R. Madhavan, K. Fregene, and L. E. Parker, "Distributed cooperative outdoor multirobot localization and mapping," *Autonomous Robots*, vol. 17, pp. 23–39, 2004.

[7] C. Estrada, J. Neira, and J. Tardós, "Hierarchical slam: Real-time accurate mapping of large environments," *IEEE Trans. on Robotics*, vol. 21(4), pp. 588–596, 2005.

[8] J. Sola, A. Monin, M. Devy, and T. Vidal-Calleja, "Fusing monocular information in multicamera slam," *IEEE Trans. on Robotics*, vol. 24(5), pp. 958–968, 2008.

[9] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state lters for view-based slam," *IEEE Trans. on Robotics*, vol. 22(6), pp. 1100–1114, 2006.

[10] H. S. Hu Gibson and D. Gamini, "3d i-slsjf: A consistent sparse local submap joining algorithm for building large-scale 3d map," in *Proceedings of the 48th IEEE Conference on Decision and Control*, Dec 2009, pp. 6040–6045.

[11] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *Int. Journal of Robotics Research*, vol. 25 (12), 2006.

[12] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27(3), pp. 418–433, 2005.

[13] T. Féraud, P. Checchin, R. Aufrère, and R. Chapuis, "Communicating vehicles in convoy and monocular vision-based localization," in *Proc. 7th IFAC Symp. on Intelligent Autonomous Vehicles (IAV2010)*, Lecce, Italy, 2010.

[14] A. Vedaldi, J. Hailin, P. Favaro, and S. Soatto, "Kalmansac: Robust filtering by consensus," in *Proc. Int. Conf. on Computer Vision*, Beijing, China, 2005.

[15] V. Delsart and T. Fraichard, "Tiji, a generic trajectory generation tool for motion planning and control," in *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2010, pp. 1439–1444.

[16] H. Strasdat, J. Montiel, and A. Davison, "Real-time monocular slam: Why filter?" in *IEEE Int. Conf. on Robotics and Automation*, Anchorage, USA, 2010.

Eigen analysis and gray alignment for shadow detection applied to urban scene images

Tiago Souza, Leizer Schnitman and Luciano Oliveira
Intelligent Vision Research Lab, CTAI
Federal University of Bahia
Salvador, Bahia, Brazil
{tiago.souza,leizer,lrebouca}@ufba.br

***Abstract*— Urban scene analysis is very useful for many intelligent transportation systems (ITS), such as advanced driver assistance, lane departure control and traffic flow analysis. All these systems are prone to any kind of noise, which ultimately harms system performance. Considering shadow as a noise problem, this may represent a critical line between the success or fail of an ITS framework. Therefore, shadow detection usually provides benefits for further stages of machine vision applications on ITS, although its practical use usually depends on the computational load of the detection system. To cope with those issues, a novel shadow detection method, applied to urban scenes, is proposed in this paper. This method is based on a measure of the energy defined by the summation of the eigenvalues of image patches. The final decision of an image region to contain a shadow is made according to a new metric for unsupervised classification called here as gray alignment. The characteristics of the proposed method include no supervision, very low computational cost and mathematical background unification, which turns the method very effective. Our proposed approach was evaluated on two public datasets.**

I. INTRODUCTION

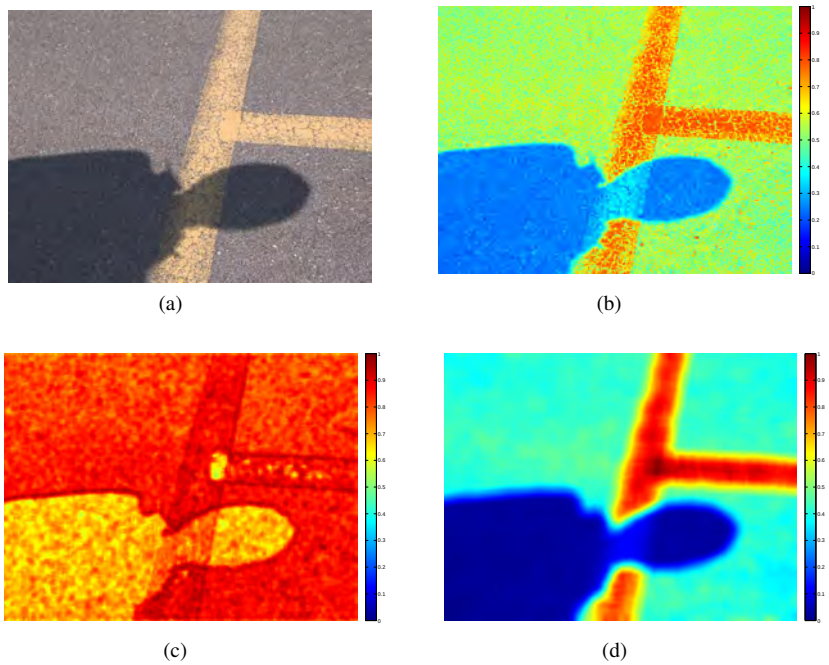
While we need light to see the world, shadows come to our eyes as an inevitable effect. When light casts shadow over objects in a scene, shadow projections (umbra and penumbra) on an image plane can bring either useful or useless information. In the field of intelligent transportation systems (ITS), as the aim is to develop advanced machine vision applications, such as video surveillance, advanced driver assistance or traffic flow analysis, shadow detection usually provides benefits for further stages of ITS frameworks. Just to cite an example, object detection systems (ODS) rely on features and classifiers in order to say where an object is located in an image [1]; in this case, shadows can harm ODS work, leading them to take shadows of objects as objects of interest.

Recognizing image shadows still remains an open and extremely challenging problem. In ITS field, we can cite that Prati et al. [2] have classified existing approaches to detect shadows in images as statistical and deterministic, according to the nature of the classification algorithms. After [2], algorithms which rely solely on statistical methods can still be categorized as parametric [3] or non-parametric [7], while the deterministic ones are based on model [8] or non-model [9]. Although in [2], the goal was to compare shadow detection methods based on moving techniques (i.e., applied on videos, and taking the temporal information in favor), the proposed taxonomy fully spans all the categories of classification methods used so far, whether on video or still images. There are two points worth noting here: we are particularly interested on still natural scene images, exploring spatial information rather than temporal one; and, although our method was conceived by means of a non-supervision framework, (and, then, it would be classified as a non-model based, deterministic approach), it cannot be seen as a learning driven method. Conversely, the proposed method described here can be described as a unified mathematical-grounded method to recognize shadows, in a very fast way. Particularly in traffic flow analysis, interests go to moving cast shadows, since it can aid in a more accurate vehicle detection; this is the case of Hsieh et al. [4], who tackle the problem of vehicle occlusions, detecting and removing moving cast shadows. For a survey of moving shadow algorithms, refer to [5].

For generic shadow detection systems, recent methods pervasively use learning to classify shadow and non-shadow image pixels, trying to tackle the problem as a composition of many different features and classifiers. For example, Zhu et al. [10] proposes a conditional random field (CRF) based system to detect cast shadow in natural scenes; there, CRF is integrated to boosted decision tree classifiers, and its parameters are learnt by a Markov random field (MRF); all classifiers run over a set of feature types, with the goal of capturing all possible shadow information in grey level images. Guo et al. [11] presents a shadow detection method based on statistics of pair-wise regions; the proposed method relies on computing illumination components from color (using LAB and RGB color spaces) and texture in each region, and comparing similar and different illumination pairs; the inference over the image regions are accomplished by means of a graph-cut based optimization. Lalonde et al. [12] describes a method which explores the local cues of the image shadow; local features are used to reason over boundaries and edges, and to create a set of shadow cues to be classified by a CRF and a decision tree. All of these works use color images, but Zhu et al. and our work.

Differently from all, our work does use neither special features, nor any learning method to decide between shadow and non-shadow. Instead, our energy-based proposed method represents a unified view which detects shadows at a glance,

This work was supported by Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB), Brazil, under the grant 6858/2011.



(a) (b) (c) (d)

Fig. 1: Eigenvalue-based energy function. (a) Original image. (b) Heat map of the original image. (c) Heat map of the entropy of the original image. (d) Heat map of the summation of the eigenvalues of image patches. Note that the heat map of the eigenvalues (d) highlights shadow region in the same way that the heat map of the entropy (c) does, which demonstrates the characteristics selected by the eigenvalue-based energy function.

just using algebraic operations over image eigenvalues. For that, we compute the summation of the eigenvalues on image patches, making decisions if a pixel is shadow or non-shadow based on the similarity of its color channels. After [6], pixels owing to cast shadow regions have its RGB components inside a symmetry axis corresponding to the background; such analysis leads to a geometric interpretation to the problem and represents a weak classifier. As in [6], we have adopted a geometric analysis, which searches for similarities in the color channels in regions of low energy in the scene. We call this task as gray alignment. Following this approach, we have achieved a state-of-the-art performance over a subset of [11]'s dataset with urban natural scenes, and over a subset of [5]'s dataset (Highway I).

The motivation to use eigenvalues is twofold: i) darker objects usually have high entropy than lighter objects, however shadowed objects (which is dark for our view) present low entropy, which distinguishes them as a shadow; ii) on the other hand, entropy filters are computationally heavy, and, by computing the eigenvalues, we can save time in a robust shadow detection. In conclusion, with an eigenvalue-based energy function, one has lighting intensity and gradient analyses all at once in a unified method to spatially detect shadow. Figure 1 illustrates these ideas; in Figs 1b-1d, the lighter objects represents shadows, while hotter objects represents the rest of the image. Shadow regions is best selected in Fig. 1d.

II. Eigenvalue-based energy function

A. Definition of the proposed energy function

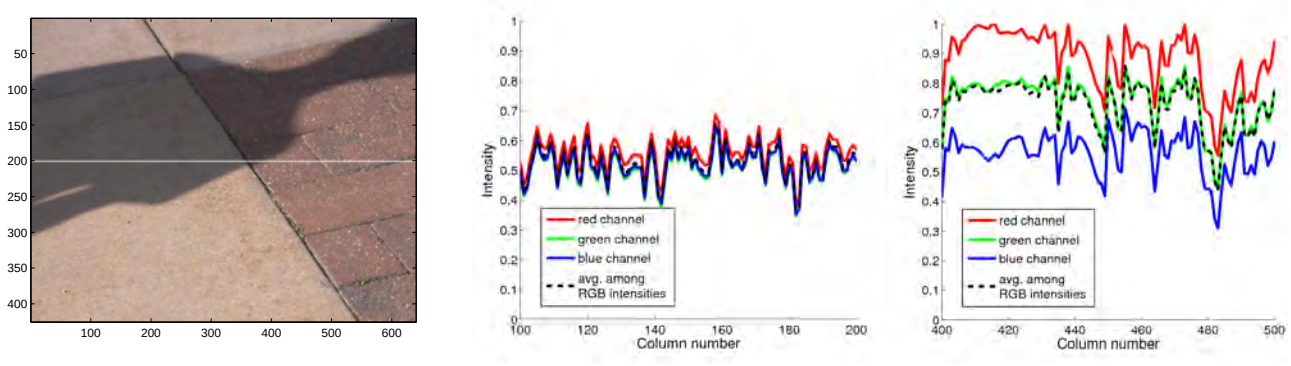
Let M be the normalized gray-level image from the RGB image I. We can state that shadow areas of M present low light intensity, small spatial gradients and low entropy, that is, they present low magnitudes with small spatial fluctuations if compared to lit image regions. Our goal is to find a function which measures the energy distribution over the image according to the characteristics above. Using that function, we will be able to segment shadow regions of M by means of the eigenvalues of the image subregions (patches). For that, we will use the Gerschgorin's circles theorem [13] below.

Theorem 1: The eigenvalues of $Q \in \mathbb{C}^{n \times n}$ are contained in the union of the n Gerschgorin's circles defined by $|z - q_{ii}| \leq r_i$, where

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} Q(i,j)$$

for $i = 1, 2, ..., n$.

After Gerschgorin[13], if a matrix presents small magnitude elements and small fluctuations in these magnitudes, so its eigenvalues also present small magnitudes if compared to a same size matrix which contradicts those preconditions. Thus, if we find relatively small subregions in M, we are able to classify such small regions as shadow or non-shadow, without the need of a spatial analysis of texture, color or



(a) (b) (c)

Fig. 2: Color intensity over the white line in the original image (a). Note the approximation among R, G, B intensities in the shadow areas in contrast to lighter image regions (c)

gradients. This is possible since we are measuring, all-at-once, the light intensity and spatial variation of that small image region. Next, we present how we define our energy-based function based on the aforementioned ideas.

Let us define S as a matrix whose elements will be calculated as

$$S_{ij} = f(\lambda_1(A^T A), \lambda_2(A^T A), ..., \lambda_n(A^T A)), \quad (1)$$

where A, of order n, is a square submatrix of M, whose central element is located on the coordinates (i, j), and f is a function of the eigenvalues λ_k, $k = 1, ..., n$, of $A^T A$ (the product $A^T A$ was used to avoid complex eigenvalues that could turn complex the computation of f). It is noteworthy that n is odd (further, this constraint will be relaxed).

If we define f as a multiplication operator among the eigenvalues of $A^T A$, then we have $S_{ij} = det(A^T A)$. In this case, if any eigenvalue of $A^T A$ is nil, S_{ij} is also nil. Furthermore, S_{ij} will have a nil value if rows or columns are equals in $A^T A$. In practice, shadows regions can imply nil eigenvalues, but it does not suffice to say that nil eigenvalues imply shadow areas. This is a common situation in very light image regions, and hence f cannot be considered as a multiplication operator.

Even knowing that the eigenvalues of $A^T A$ are all real, it is impossible to guarantee that all values are not nil, and thus S_{ij} cannot simply be a division relation among the eigenvalues of $A^T A$. A simple and coherent choice for f is a summation, since S_{ij} will be nil if and only if all eigenvalues of $A^T A$ is nil. Because of all elements of A is non-negative, the result of the summation will never be negative. According to that, we can redefine S as

$$S_{ij} = \sum_{k=1}^{n} \lambda_k(A^T A). \quad (2)$$

Note that this new S can have a high computational cost to be computed, since all operations are pixel-wise over M. This high cost becomes inevitable for high values of n. To solve this problem, let us define $A_{n\times n} = (v_1\ v_2\ \dots\ v_n)$ where $v_k = (a_{1k}\ a_{2k} \dots\ a_{nk})^T \in \mathbb{R}^n$ and $k = 1, 2, \dots, n$. Then, we have

$$A^T A = \begin{pmatrix} v_1^T v_1 & v_1^T v_2 & \dots & v_1^T v_n \\ v_2^T v_1 & v_2^T v_2 & \dots & v_2^T v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n^T v_1 & v_n^T v_2 & \dots & v_n^T v_n \end{pmatrix}. \quad (3)$$

From (3), we can write S as

$$\begin{aligned} S_{ij} &= Trace\left(A^T A\right) \\ &= \sum_{k=1}^{n} v_k^T v_k \\ &= \sum_{k=1}^{n} \|v_k\|_2^2. \end{aligned} \quad (4)$$

Following this approach, let A be a subregion of M, centered in a pixel located in (x, y) and $Q = A^T A$ (with that, we guarantee that all eigenvalues of Q are real). In order to increase the precision of our detector, we have associated the summation of the eigenvalues of Q to a temporary matrix E, defined as

$$\begin{aligned} E_{xy} &= \sum_{k=1}^{n} \lambda_k(Q) \\ &= \sum_{k=1}^{n} \lambda_k(A^T A) \end{aligned} \quad (5)$$

B. Gray alignment-based unsupervised classifier

Figure 2 illustrates the phenomenon of shadow occurrence on a surface. Note that each RGB channel over a pixel becomes closer to the mean of RGB intensities, that is, there is an alignment between the color vector ($\vec{C}$) and mean color ($\vec{p}$) defined as

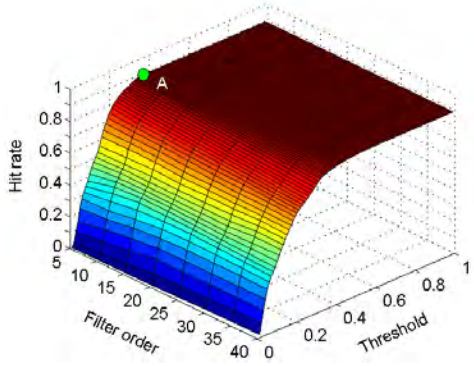


Fig. 3: Determining the best threshold in function of the filter order (point A, $threshold$ = 0.39 and matrix order of 5).

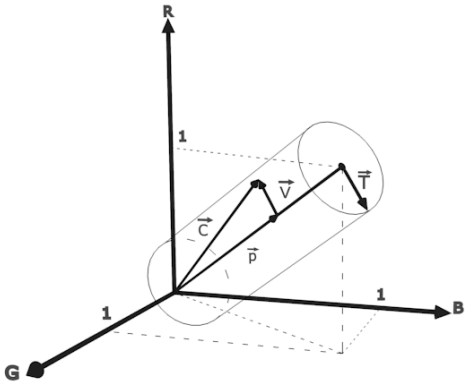


Fig. 4: Geometric interpretation of gray alignment. Any point out of the cylinder of radius equal to $\vec{T}$ is automatically deemed as non-shadow.

$$\vec{C} = R\hat{r} + G\hat{g} + B\hat{b} \tag{6}$$

$$\vec{p} = \mu(\hat{r} + \hat{g} + \hat{b})\,, \tag{7}$$

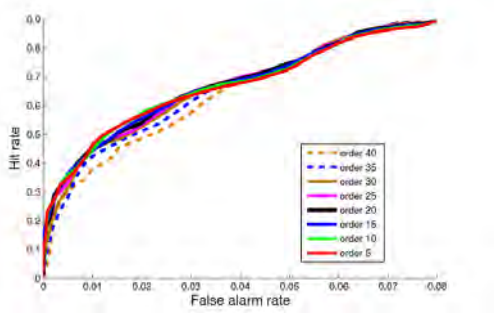
where

$$\mu = \frac{R+G+B}{3}\,. \tag{8}$$

This is equivalent to a saturation reduction of the color in badly lit regions. Figure 4 depicts the geometric interpretation of the gray alignment principle, indicating a cylinder around the line $R = G = B$, where out of that, no combination of colors indicates that a pixel belongs to a shadow area. Still according to Figure 4, we have that $I(x, y)$ belongs to a shadow area if, and only if, the associated color to $I(x, y)$ is contained inside the cylinder of radius equal to $|\vec{T}|$. Algebraically, it is given by

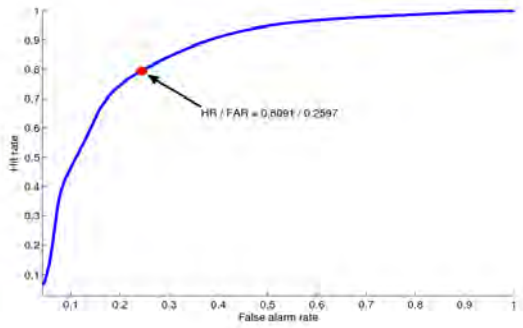
$$\vec{c}_{shadow} \Longrightarrow |\vec{V}| = |\vec{C} - \vec{p}| < |\vec{T}| \tag{9}$$

In order to minimize the computational load of Eq. (9), we rewrote it as



(a)

Fig. 5: ROC curves on a subset of [10]'s dataset. There is no significant impact on the detection performance by varying the filter order. In FAR = 7%, HR = 89%.



(a)

Fig. 6: ROC curves on a subset of [5]'s dataset (Highway I). The operating point corresponds to 80.91% of hit rate (HR) and 25.97% of false alarm rate (FAR).

$$|R \cdot G \cdot B - \mu^3| < |\vec{T}|\,, \tag{10}$$

where the operation in the equation is pixel-wise.

For a pixel to be considered within a shadow area, it needs to attend the required conditions of the gray alignment classifier, and must have its energy level bellow a certain value, according to the eigenvalue-based energy function. Based on our experimental analyses, the best threshold for the eigenvalue-based function was found to be 0.39 and $|\vec{T}|$ was 0.005.

III. Result Analysis

Our method requires just two parameters: the order n of the sub-matrices and the threshold to binarize the image as shadow or non-shadow. To determine the best threshold to our classifier, the surface in Figure 3 was built by varying the filter order (equivalent to the size of the image patch to calculate the eigenvalues) and the threshold, in the interval $[0, 1]$, of the classifier defined in 10. This is found in point A in the figure.

To evaluate the performance of the proposed system, two datasets were used: a subset of [10]'s dataset with 39 images of urban scenes, and a subset of [5]'s dataset with 80 images.

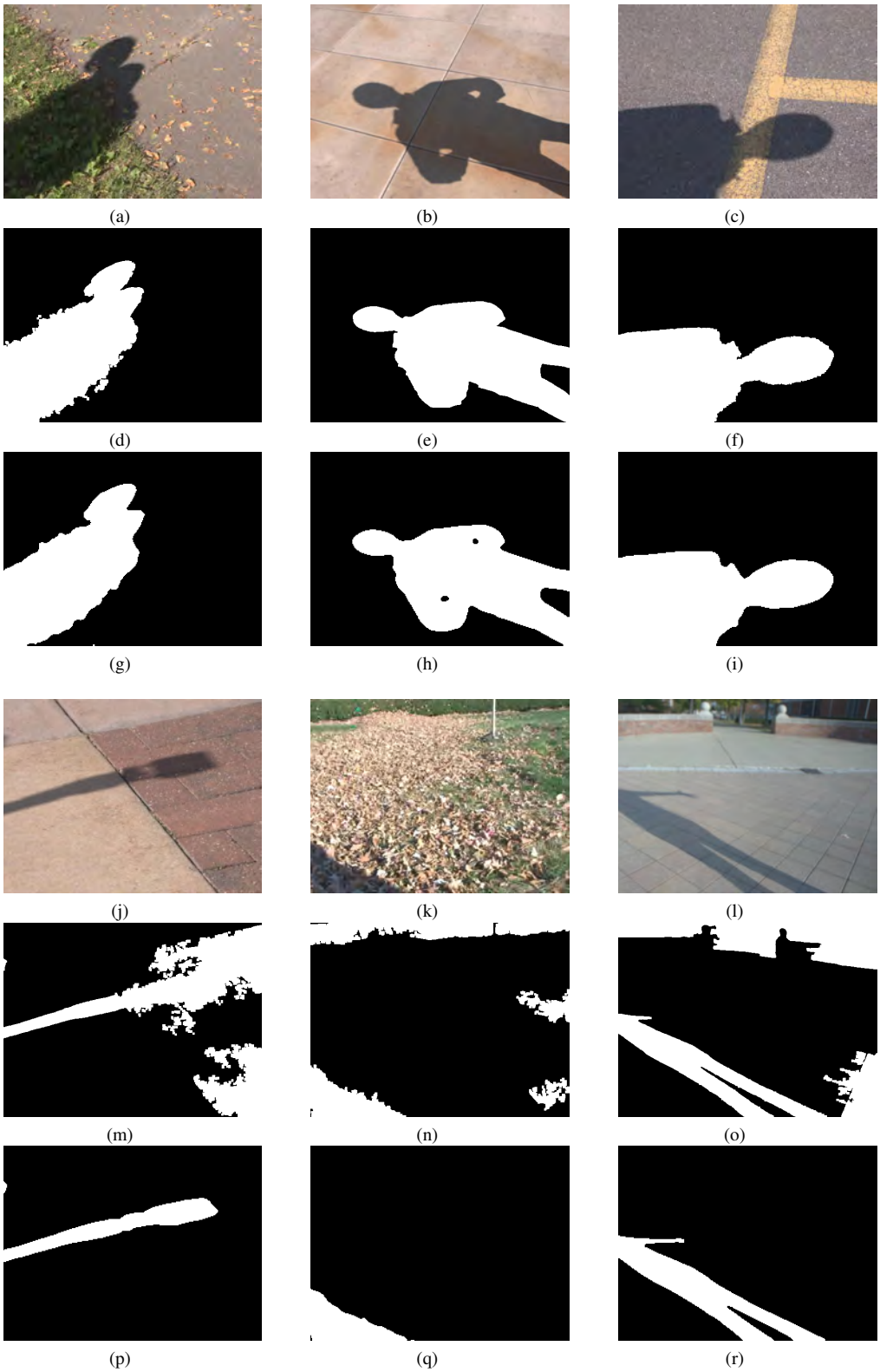


Fig. 7: Detection examples. **(a-i) near perfect detections:** First line (a-c), original images; second line (d-f), detection results; third line (g-i), ground truth. **(j-r) bad detections:** First line (j-l), original images; second line (m-o), detection results; third line (p-r), ground truth.

Shadow detection	Shadow	Non-shadow
Shadow (GT)	**0.7498**	0.0838
Non-shadow(GT)	0.2502	**0.9162**

TABLE I: Confusion matrix of the detector with approximately 75% of shadow detection and 92% of non-shadow detection.

Figure 5 shows the ROC curve of our shadow detector over [10]'s dataset. It is noteworthy that even varying the eigenvalue matrix order (filter order), there was no significant improvement in the detection performance. It indicates that a lower matrix order can be more suitable in terms of computation speed, since it reduces significantly the number of operations to compute the summation of the eigenvalues. Table I summarizes the confusion matrix with approximately 75% of shadow detection and 92% of non-shadow detection over the [10]'s dataset.

Figure 6 shows the ROC curve of the proposed method over a subset of [5]'s dataset (Highway I). The annotation was remade since we did not find the original one. In view of that, only the cast shadows (or those ones completely projected on the ground) were considered. One important fact which came with that choice was the method to make mistakes over the shadows detected over the vehicles (see Fig. 8). Even though, it was achieved approximately 81% of hit rate.

In Figure 7, there are some examples of the resulting images after the detection over [10]'s dataset: Figures 7 (d)-(f) show some near perfect results, while Figures 7 (m)-(o) show bad results. The bad results can be explained by the detection of some penumbras in some sparse regions of the image which were not annotated. In Figure 8, some resulting images after performing the method over the subset of [5]'s dataset are depicted. Annotation of the images of Highway I's dataset was performed considering only the cast shadow, and it is noteworthy that shadows on the cars (form shadows) represent detection mistakes (in practice, it is something should also be removed as shadow).

IV. Conclusion

This paper has presented a novel method for image shadow detection, based on eigenvalue-based energy function and gray-level alignment classification. The motivation of the use of eigenvalues over a matrix of the kind $A^T A$, where A is an image patch, is grounded on the fact that: i) darker objects usually have high entropy than lighter objects, although shadowed objects (which is dark in the image) presents low entropy, which distinguishes them as a shadow; ii) entropy filters are computationally heavy, and, by computing the eigenvalues, we can save time in a robust shadow detection; an eigenvalue-based energy function presents then a clear distinction between lighting intensity and gradient information all at a time, providing a unified method to spatially detect shadow. Our proposed method have demonstrated to own very low computational cost, to be unsupervised, and to perform very efficiently over a subset of a public dataset.

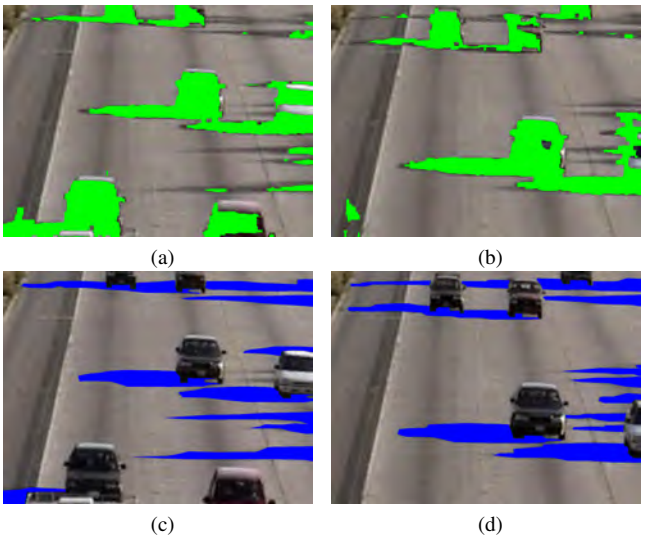


Fig. 8: Results on [5]'s dataset (Highway I). First line (a-b), shows detection results, while second line (c-d) shows the annotations.

Future work goes toward the integration of the method in a temporal framework.

References

[1] Oliveira, L.; Nunes, U.; Peixoto, P., On exploration of classifier ensemble synergism in pedestrian detection. IEEE Transactions on Intelligent Transportation Systems, pp. 16–27, 2010.

[2] Prati, A.; Mikic, I.; Cucchiara, R.; Trivedi, M. M., Comparative evaluation of moving shadow detection algorithms. In: IEEE CVPR workshop on Empirical Evaluation Methods in Computer Vision, 2001.

[3] Mikic, I.; Cosman, P. C.; Kogut, G. T.; Trivedi, M. M., Moving shadow and object detection in traffic scenes. In: International Conference on Pattern Recognition, vol 1, pp. 321-324, 2000.

[4] Jun-Wei, H.; Shih-Hao, Y.; Yung-Sheng, C.; Wen-Fong, H., Automatic traffic surveillance system for vehicle tracking and classification, In: IEEE Transactions on Intelligent Transportation Systems, vol. 7, issue 2, pp. 175–187, 2006.

[5] Prati, A.; Mikic, I.; Trivedi, M.; Cucchiara, R., Detecting Moving Shadows: Algorithms and Evaluation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, issue 7, pp. 918–923, 2003

[6] Huang, J-B; Chen, C-S, Learning Moving Cast Shadows for Foreground Detection, In: International Workshop on Visual Surveillance, 2008.

[7] Haritaoglu, I.; Harwood, D.; Davis, L. S., W4: Real-time surveillance of people and their activities. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 809–830, 2000.

[8] Onoguchi, K., Shadow elimination method for moving object detection. In: International Conference on Pattern Recognition, vol. 1, 583–587, 1998.

[9] Stander, J.; Mech, R.; Ostermann, J., Detection of moving cast shadows for object segmentation. IEEE Transactions on Multimedia, 65-76, 1999.

[10] Zhu, J.; Samuel, K.; Masood, S.; Tappen, M., Learning to recognize shadows in monochromatic natural images. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 223–230, 2010.

[11] Guo, R.; Dai, Q.; Hoiem, D., Single-image shadow detection and removal using paired regions. In: IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[12] Lalonde, J. F.; Efros, A. A.; Narasimhan, S. G., Detecting ground shadows in outdoor consumer photographs. In: European Conference on Computer Vision, pp. 322-335, 2010.

[13] Gerschgorin, S., Über die abgrenzung der eigenwerte einer matrix, In: Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na., no. 6, pp. 749–754, 1931.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session IV

Navigation, Control, Planning

- **Keynote speaker: Rüdiger Dillmann (KIT, Karlsruhe, Germany)**
  Title: Interpretation of Situative Sensor-Data and Continuous Decision Making for Cognitive Automobiles
  Co-Authors: S. Brechtel, T. Gindele

- **Title: An Efficient Heuristic Estimate for Non-holonomic Motion Planning**
  Authors: J.W. Choi

- **Title: Short term path planning using a multiple hypothesis evaluation approach for an autonomous driving competition**
  Authors: M. Oliveira, V. Santos and A.D. Sappa

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session IV

Keynote speaker: **Rüdiger Dillmann (KIT, Karlsruhe, Germany)**

Interpretation of Situative Sensor-Data and Continuous Decision Making for Cognitive Automobiles

Co-Authors: S. Brechtel, T. Gindele

Abstract : Autonomous automobiles must be capable to decide continuously on their adequate behaviour in a highly dynamic environment. Usually such vehicles are supported with uncomplete and noisy perceptive data with different sensor modalities. Probabilistic and predictive methods can be applied to estimate the actual situationand the intension of other traffic agents around the autonomous vehicle and to generate the best and safe behaviour. Predictive driving requires understanding the sensorial observations and the behaviour of the other vehicles in standard and non standard traffic situations. With such a background, structural bootstrapping towards not befor seen situations can be realized and logically explained with the help of learning strategies. A probate method is to collect automatically experience from traing data and online observations in form of Bayesian Nets. For processing of the consequences of actions under uncertainties continuous POMDPs may be suitable to generate adequate decisions. Planning algorithms, decision problems and its implementation and experimental verification as well as actual research results are presented in this talk.

Biography: Prof. Rüdiger Dillmann received his PhD in Electrical Engineering in 1980 from the University of Karlsruhe. Since 1987 he is Professor at the Department of Computer Science and Engineering at the Karlsruhe Institute of Technology and in 2009 he also became the Director of the Humanoids and Intelligence Systems Research Lab. From 2002 -2010 he was President of the Research Center for Information Science (FZI), Karlsruhe. As a leader of two institutes Prof. Dillmann supervises several research groups in the areas of humanoid robotics with a special interest on intelligent, autonomous and mobile robots, machine learning, machine vision, man-machine interaction, robotics in surgery and simulation techniques. Since 2009 he is speaker of the Institute of Anthropomatics at the KIT. He is coordinator of the German Collaborative Research Center “Humanoid Robots”, Co- Editor in Chief of the journal “Robotics and Autonomous Systems” from Elsevier and Editor in Chief of the book series COSMOS from Springer.

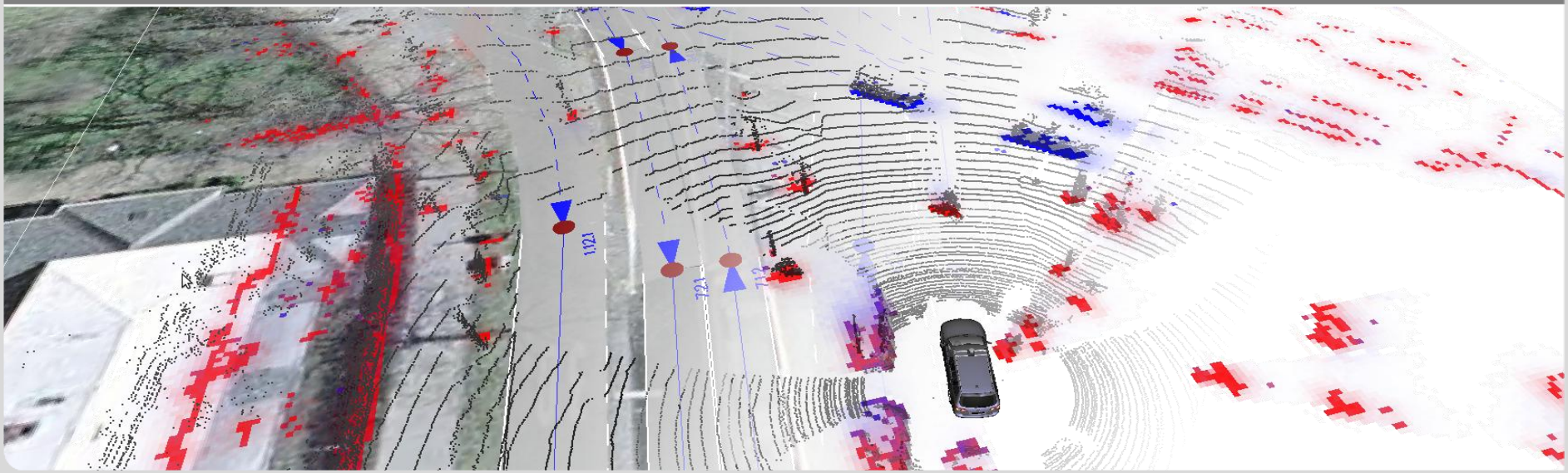
2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Karlsruher Institut für Technologie

Interpretation of Situative Sensor-Data and Continuous Decision Making for Cognitive Automobiles

R. Dillmann, T. Gindele, S. Brechtel

Humanoids and Intelligence Systems Laboratories – Institute for Anthropomatics



Introduction

- Autonomous cars have an encreasing role in transportation
  - Automated transportation systems allow new car sharing concepts
  - Cooperative Driving
  - Safety and better use of road infrastructure
  - Economical and comfortable
  - …

- In real life, traffic participants need to understand, plan, learn and to be self-aware of their perception and execution limitations

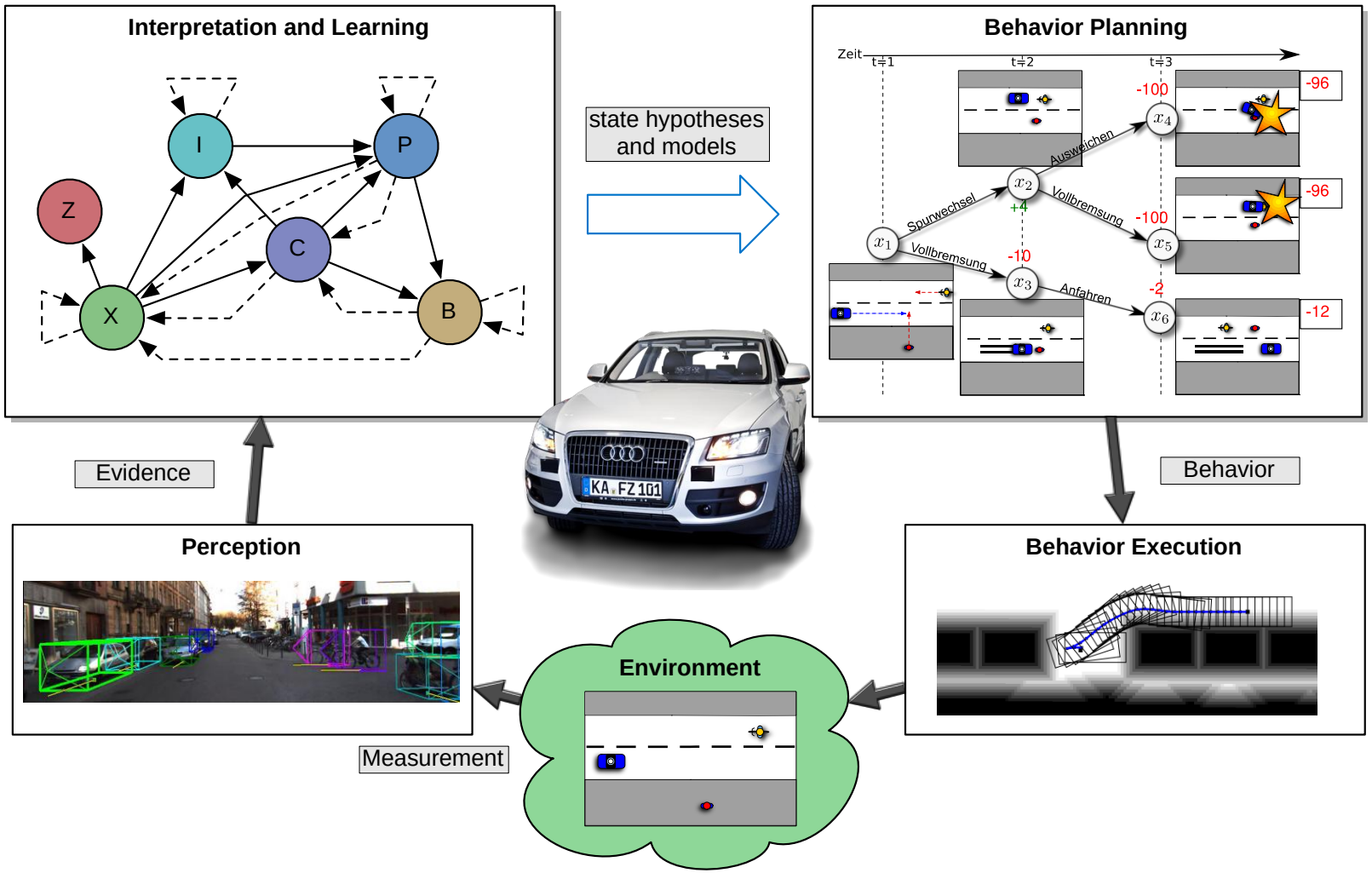
→These capabilities lead to Cognitive Car or Robotized Cars

Outline of the talk

- Cognitive Automobiles
- Perception
- Interpretation and Learning
- Decision Making
- Applications

COGNITIVE AUTOMOBILES

Architecture



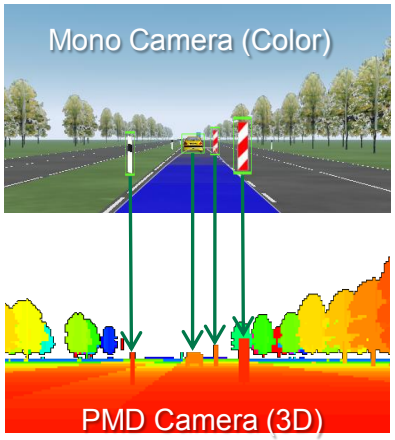
Collaborative Research at HIS and FZI

FZI Living Lab Automotive: The instrumented Test Vehicle

- Multicore Computation Systems
- Integrated Sensors
  - PMD Camera front and rear
  - Ladybug3 / Velodyne
  - Driver observation cameras 2D/3D
  - Front/Rear cameras
  - Localization system
  - 3 IBEO Laser (2 front, 1 rear)
  - Vital sensors
- Several Communication and Power Interfaces
  - 12V / 220V
  - Firewire
  - USB
  - Ethernet
  - MOST
  - CAN
- Several Displays
  - MMI Display
  - Combi Display
- Actuators for autonomous driving
- Continuous integration of algorithms

FZI Living Lab Automotive: Driving Simulator

- Driving dynamics simulation
- Generation of virtual sensor data
- 270° panoramic projection
- Integration of ADAS
- Force Feedback steering



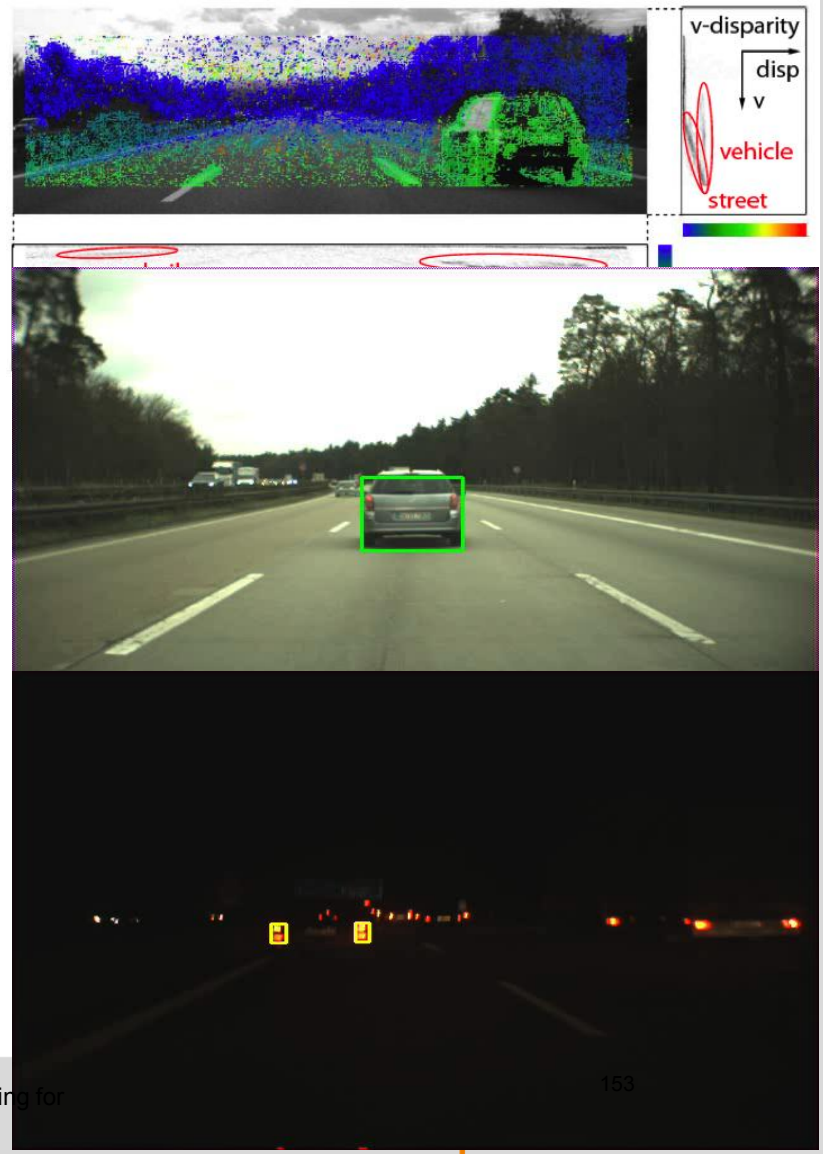
PERCEPTION

Perception of Static and Semi-Static Environment

- Sign recognition, signpost and additional signs
  - Pre-processing
  - Feature based segmentation
    - Form based, geometrical properties
    - Colour based, retro reflection and illumination based
  - Sub segmentation of complex signs
  - Pictogram classification by cascaded SVM
  - Text recognition
  - Tracking and temporal fusion
- Relevance estimation Assignment to lanes
  - Fusion with lane tracking
  - Using a-priori knowledge about signs
  - Probabilistic approach (Mixture of Gaussian)

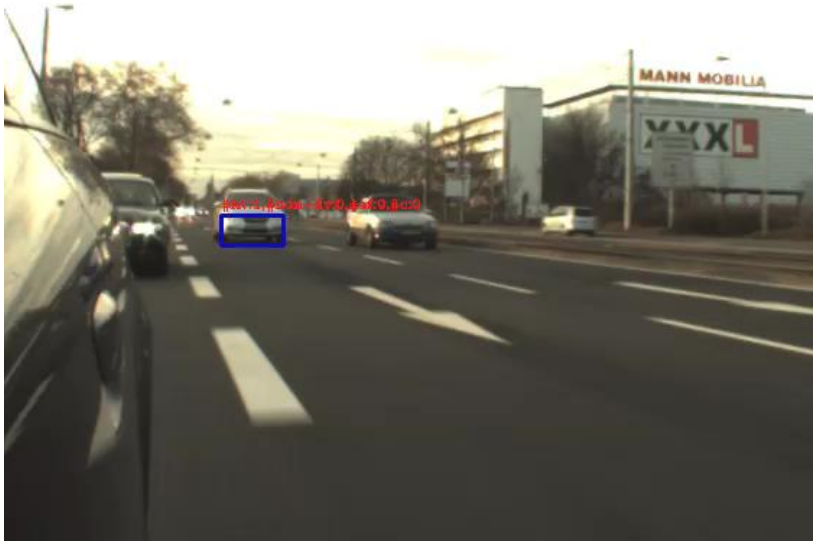
Perception of Dynamic Objects

- Combining different sensor systems
  - 3D TOF, 2D Video, Radar, Stereo
  - 3D segmentation (U/V disparity or TOF),
  - 2D detection by symmetry, shadow
  - Viola Jones detector
  - Gabor-features and SVM classification
  - Kalman tracker
- Detection during dusk and at night
  - Adaptive Gauss filtering
  - Hypothesis generation
  - Hypothesis verification by classification (coming soon)
- Manoeuvre state detection
  - Breaking , lane changing



Perception of Dynamic Objects

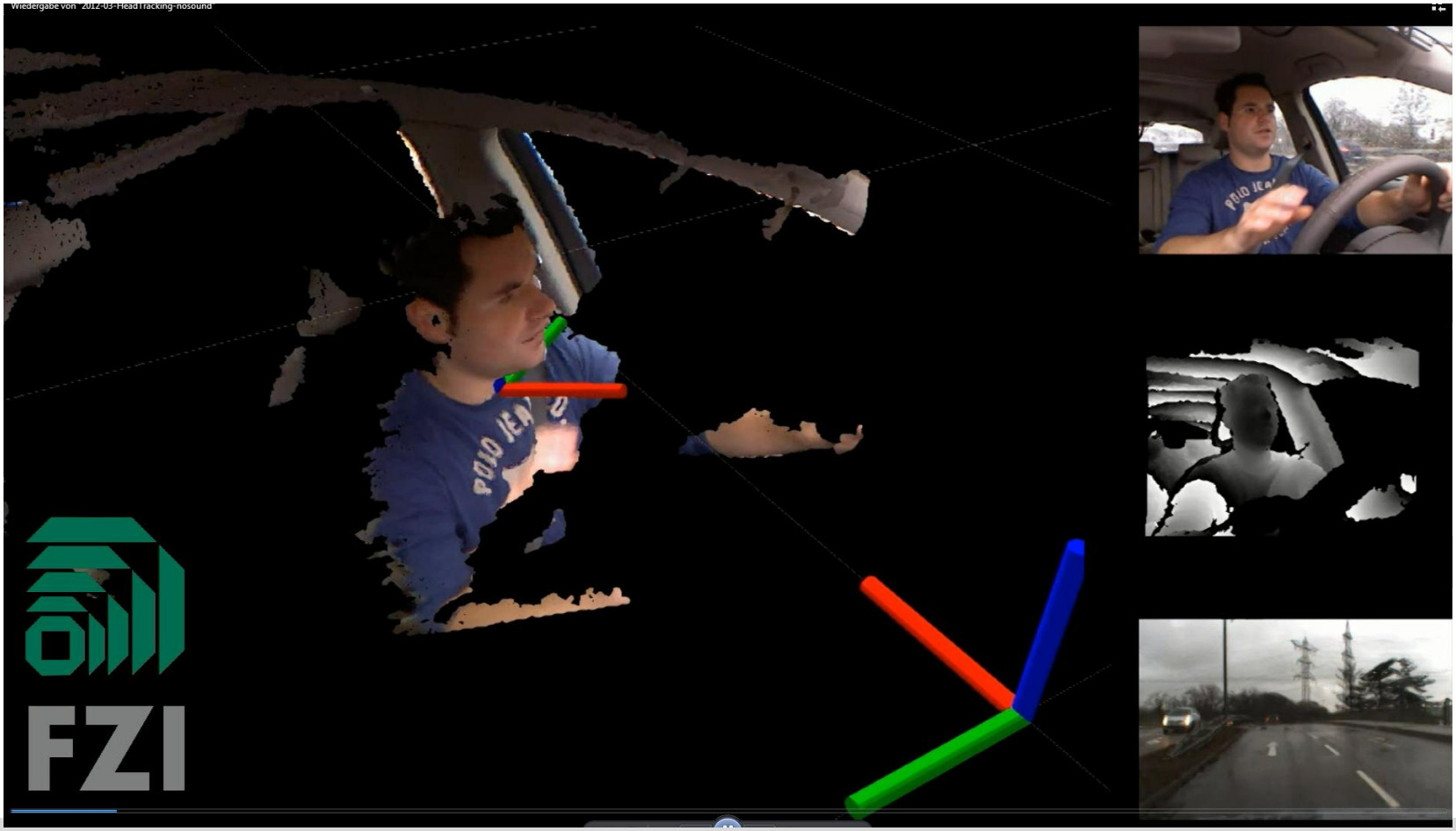
- Blind spot detection
  - Appearance based hypothesis generation (symmetry, shadow,…)
  - HOG feature based hypothesis verification (classifcation)
  - Optical flow (near field tracking)



- Pedestrian detection
  - PMD based segmentation
  - Vision based – Ada boost classification
  - Tracking

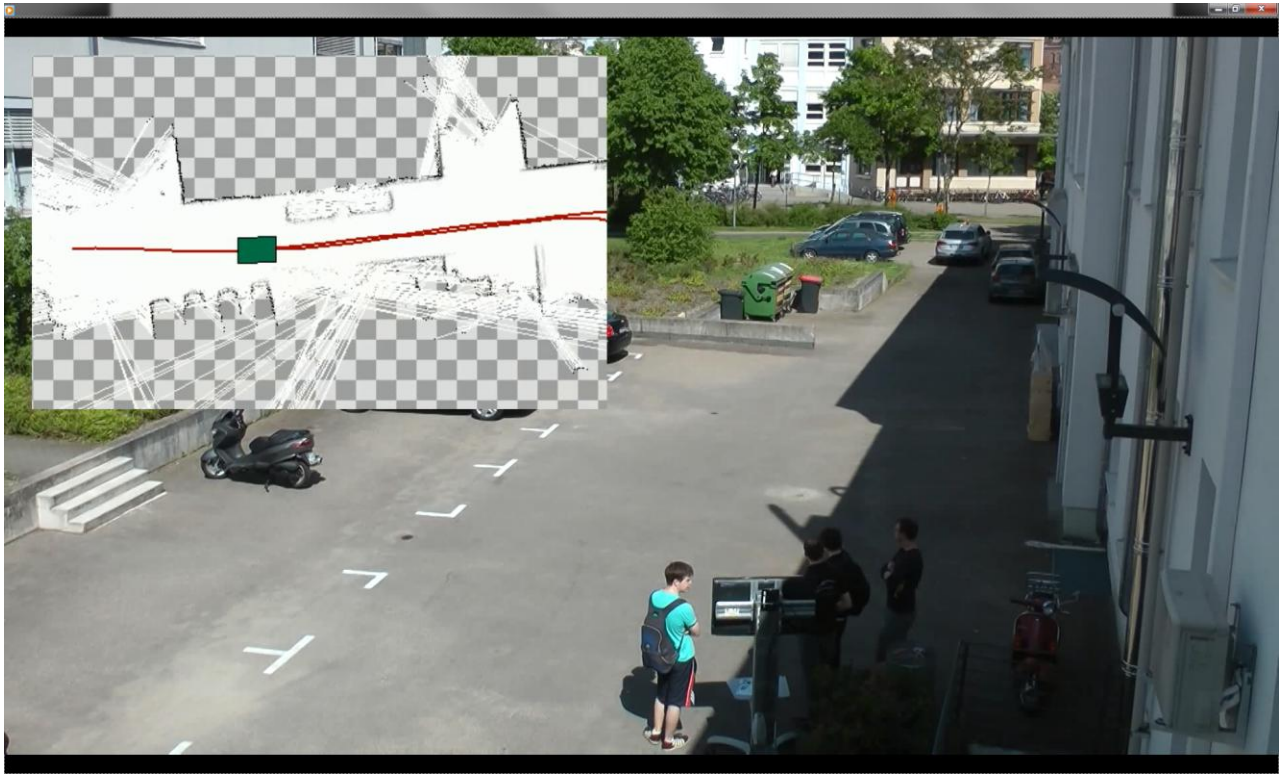
Driver Observation

- Tracking the driver‘s head movement



Localization and Mapping

- IMU based Simultaneous Localization and Mapping
- Keeps a local 2D environment map to extend the sensor range
- Uses extended version of the *GMapping* algorithm (http://openslam.org/gmapping.html)



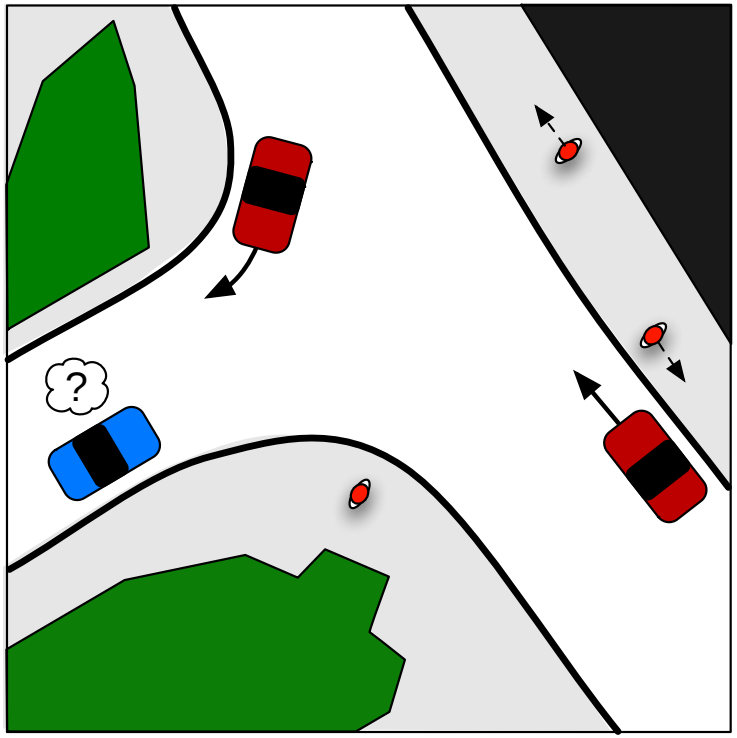
STATE ESTIMATION AND LEARNING

Situation Assessment
Requirements

- Estimation of the environment state
- Behavior recognition of traffic participants
- Situation understanding
- Anticipation

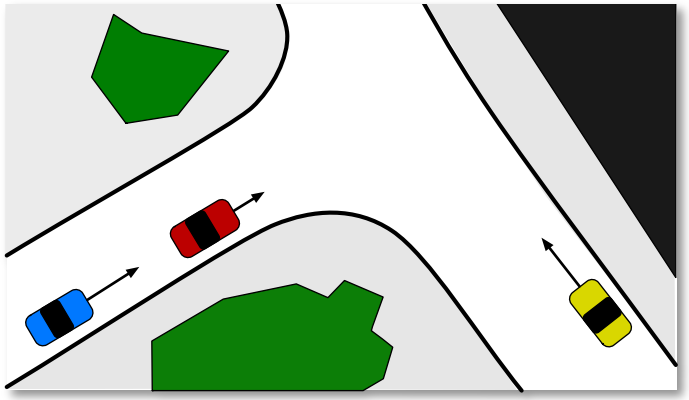
- Learning
- Robust
  - Sensor noise
  - Partial observability

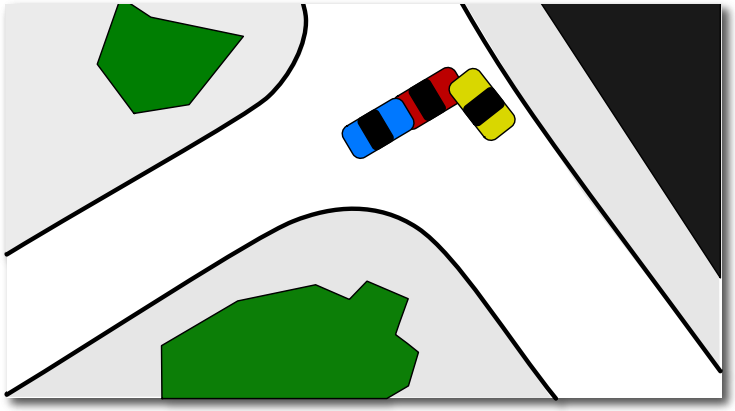
➢ Basis for Behavior Decision



Situation Assessment

- Classical approach: Instantiate independent Kalman filters for all traffic participants
- not sufficient for traffic scenarios!
- ➢ Interactions and situational context have to be considered

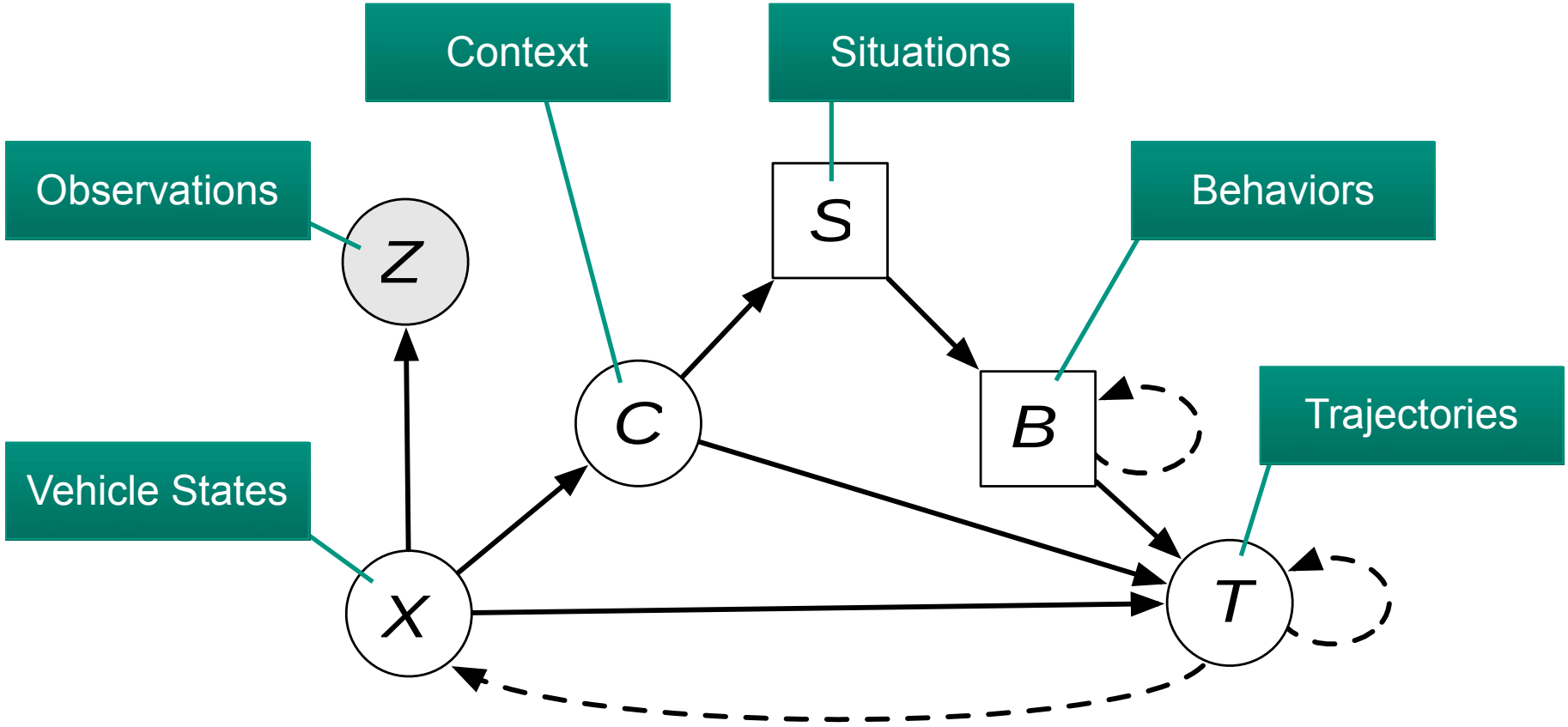




- Our approach: Reason about the decision-making of traffic participants to predict their behavior
- Situation ➡ Behavior ➡ Trajectory ➡ Next vehicle state
- Formulation as Bayesian Filter

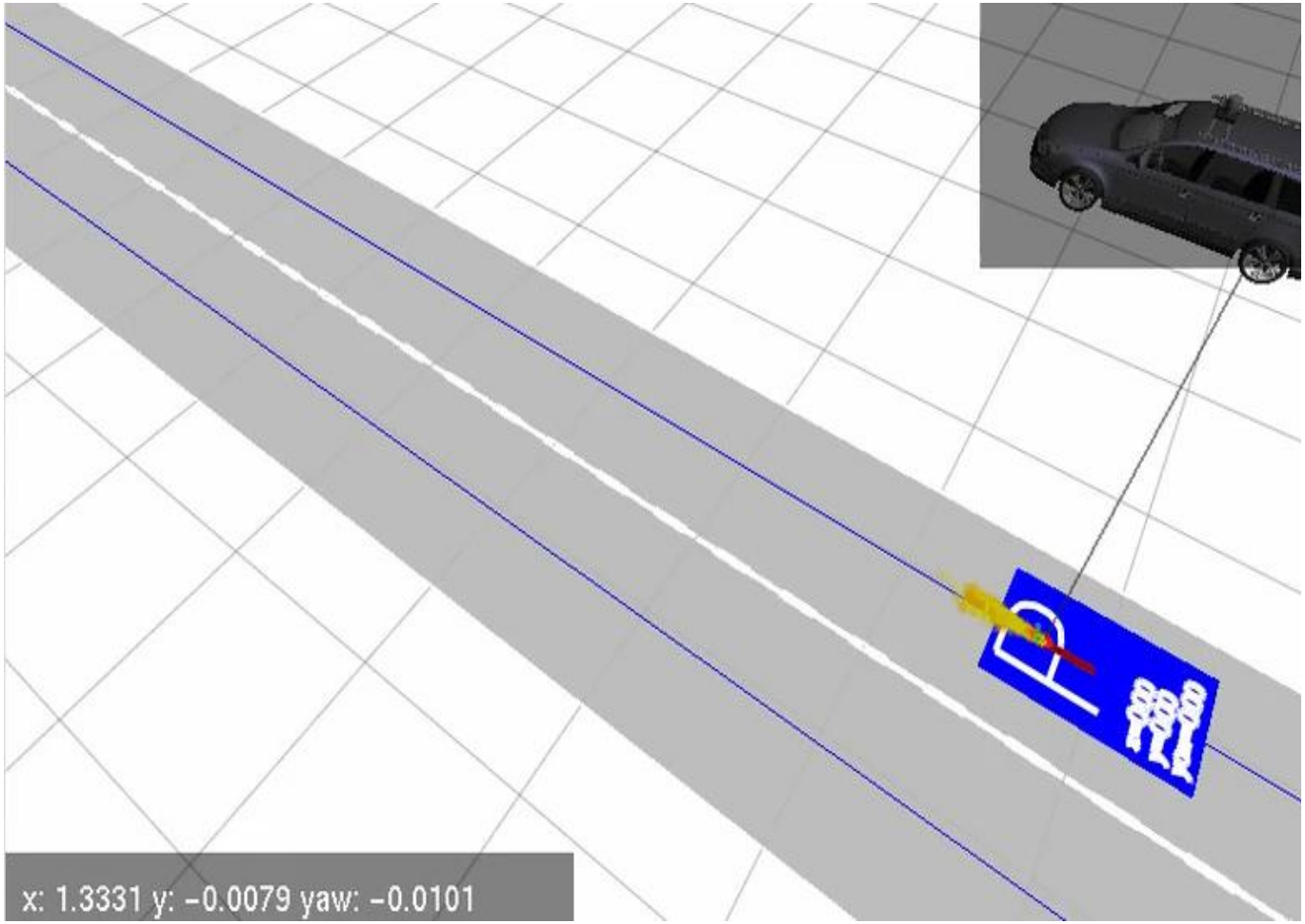
Bayesian Filter

- Dynamic Bayesian Network
- Combines symbolic and subsymbolic representations



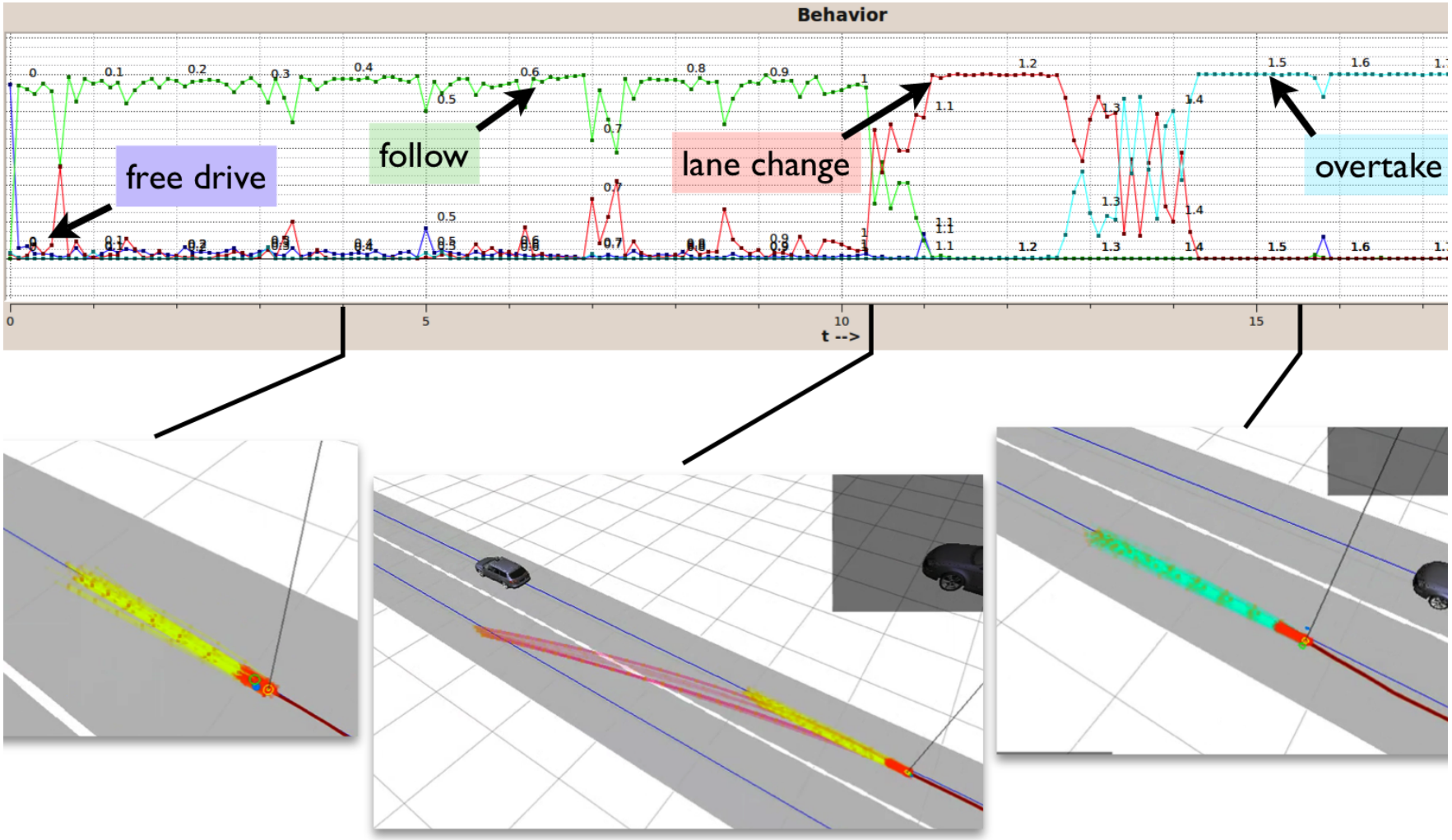
Situation Assessment

Behavior and Trajectory Estimation



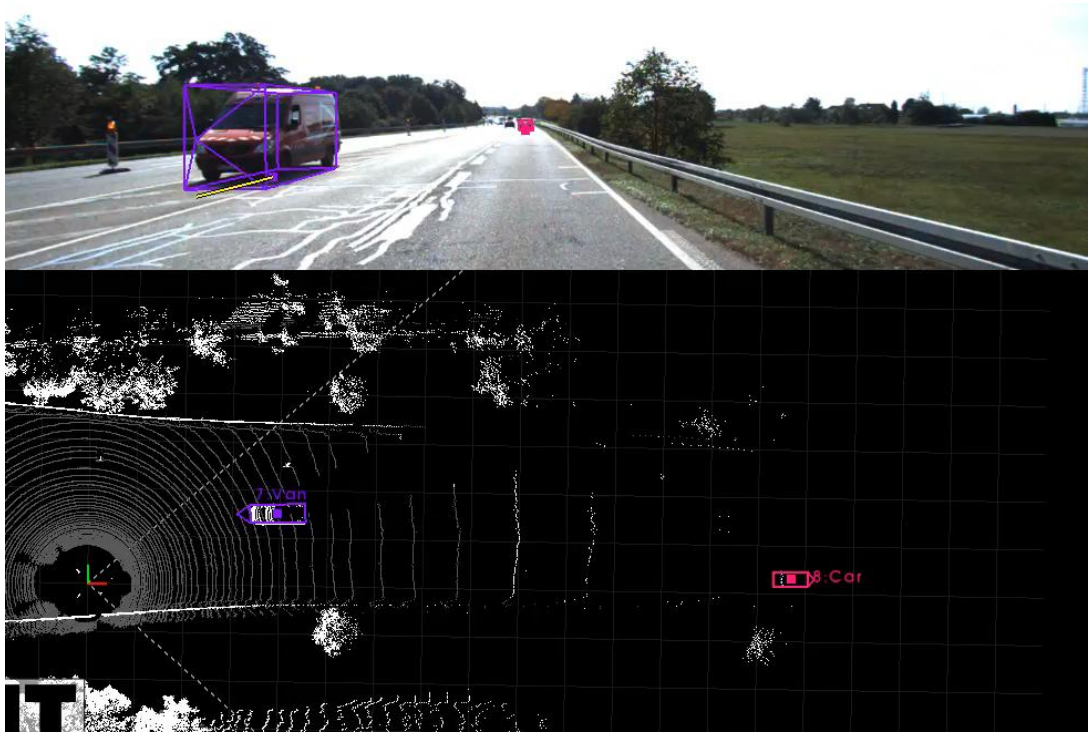
- Particles
- Estimation
- Ground Truth
- Measurement

Situation Assessment
Behavior Estimation



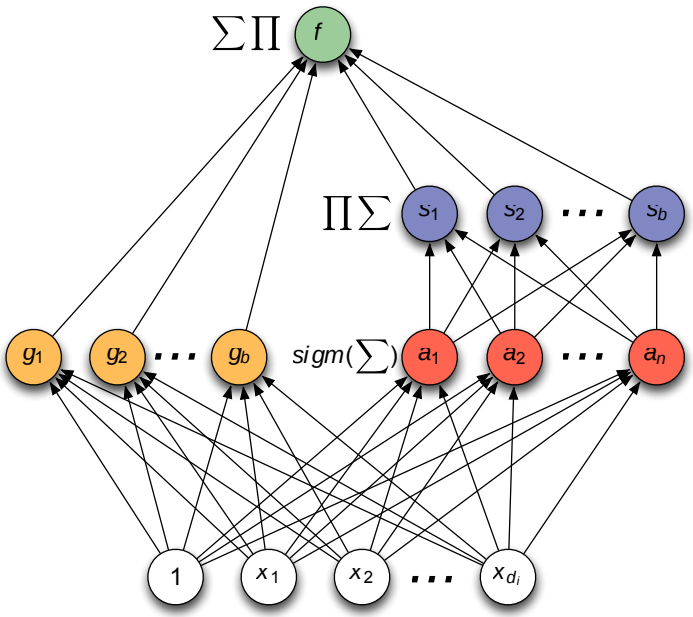
Learning of behavior models from observations

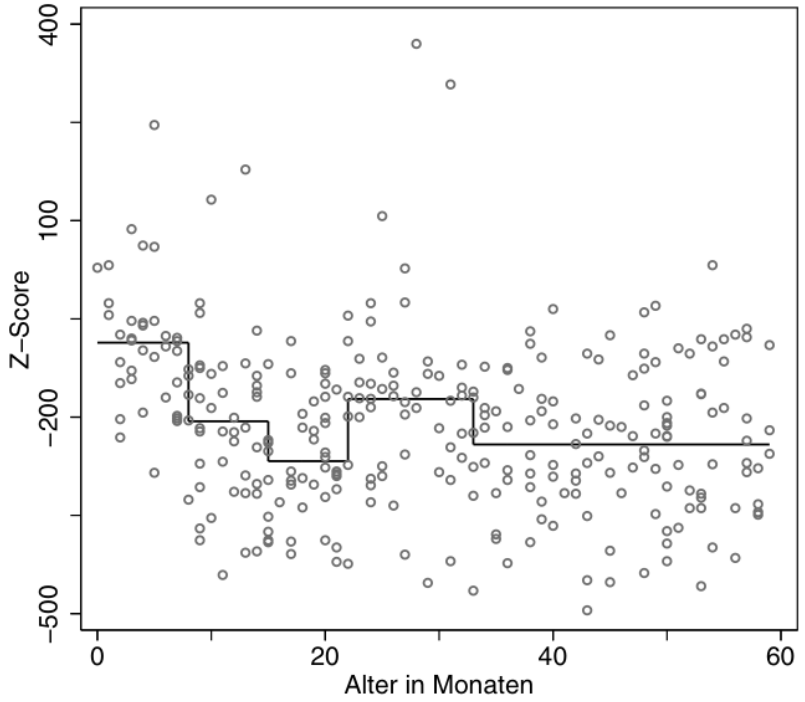
- Observation of traffic participants allow model learning
- Many training samples can be obtained
- Online learning algorithms needed to cope with data size



Source: KITTI Vision Benchmark Suite (http://www.cvlibs.net/datasets/kitti)

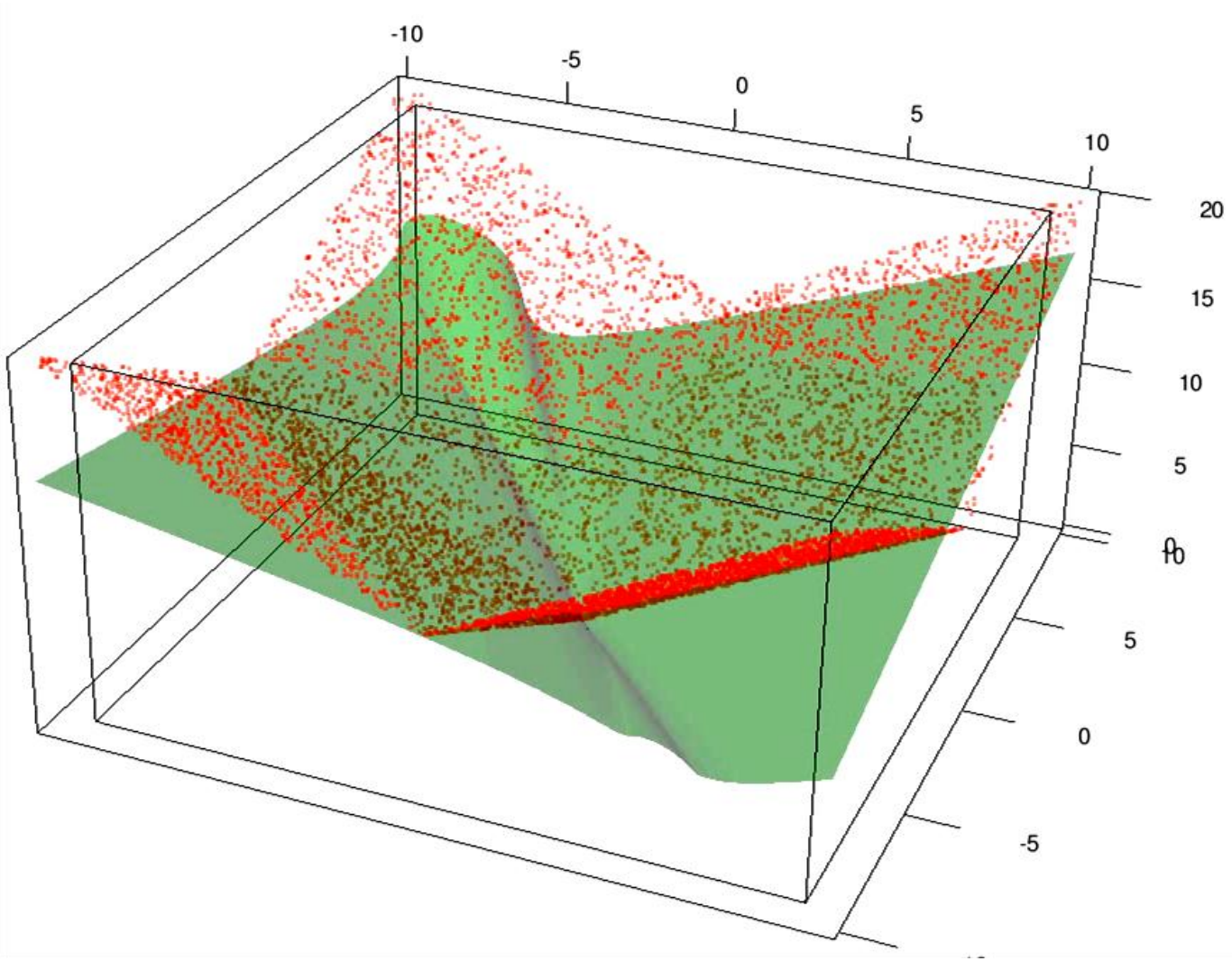
Learning Algorithm
Generalized Regression Trees

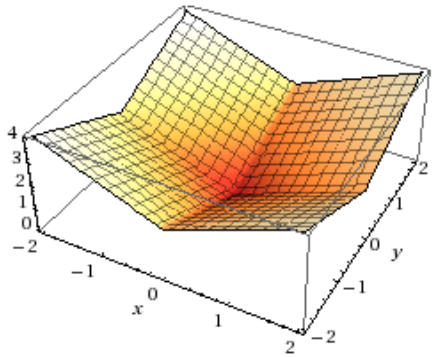




- Generalization of decision / regression trees
- Online parametric learning with stochastic gradient descent
- Extendable with random forests and deep networks

Example: Learning of a 2d function

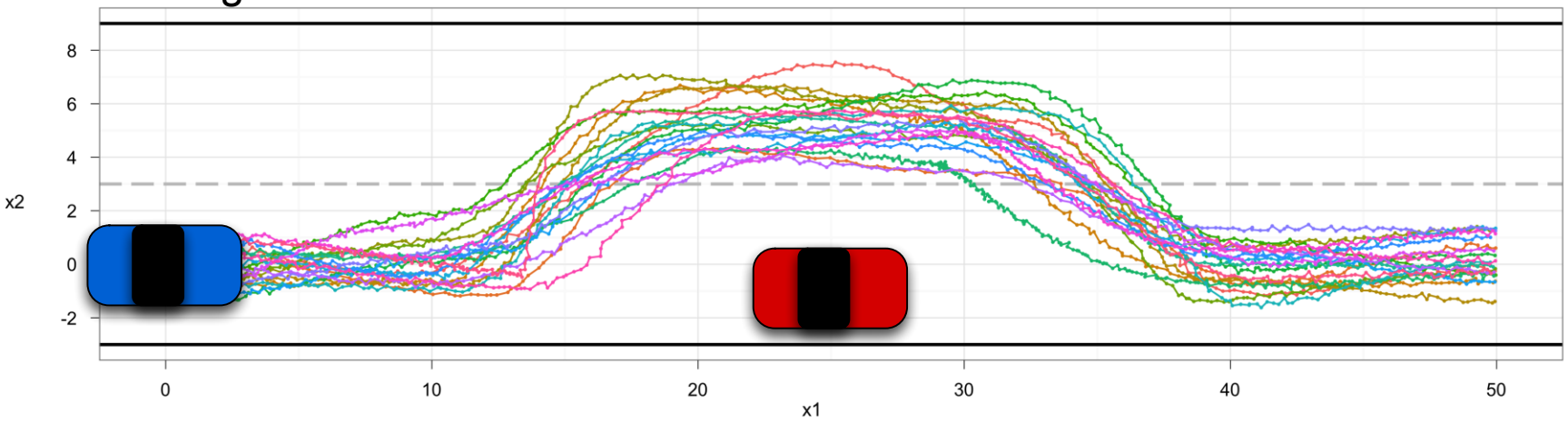




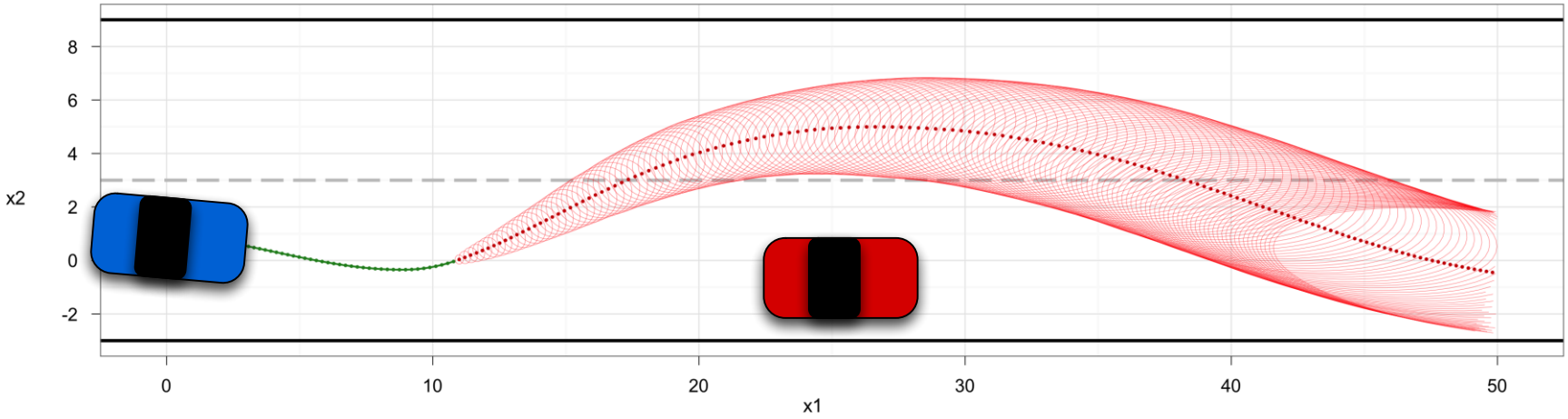
Situation Assessment

Learning of an evasive maneuver

Training Data:



Trajectory prediciton with learned model:



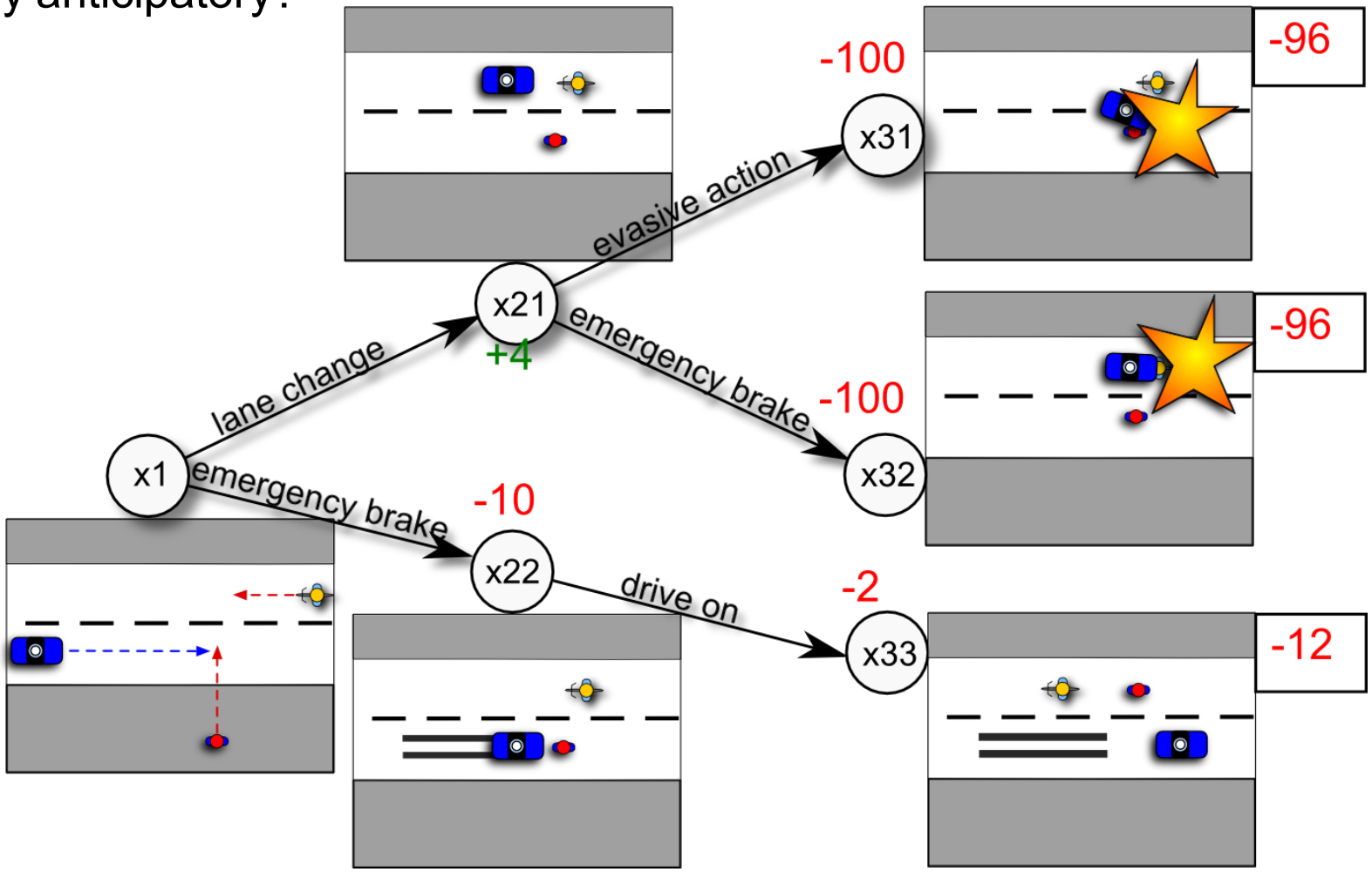
BEHAVIOR PLANNING

Symbolic Planning for Behavior Decision
Requirements

- What would one expect of strategic decision making of cognitive cars and how could it be reached?
- Anticipatory driving without error-prone manual modeling
  - ➢ Planning
- Defective and incomplete perception
  - ➢ Probabilistic Models
- Generalizability
  - ➢ Gathering experience from simulated and real drives

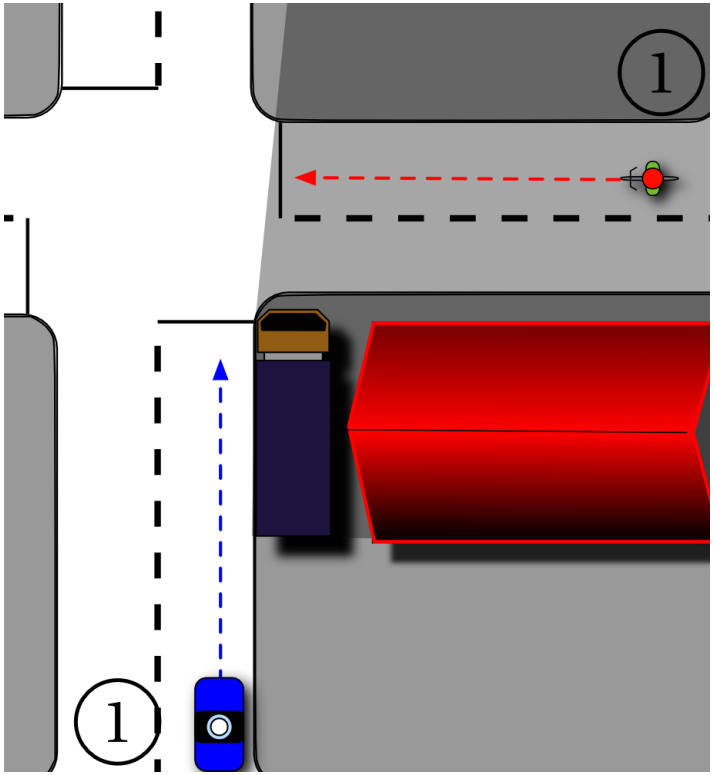
Symbolic Planning for Behavior Decision
Look Ahead Planning

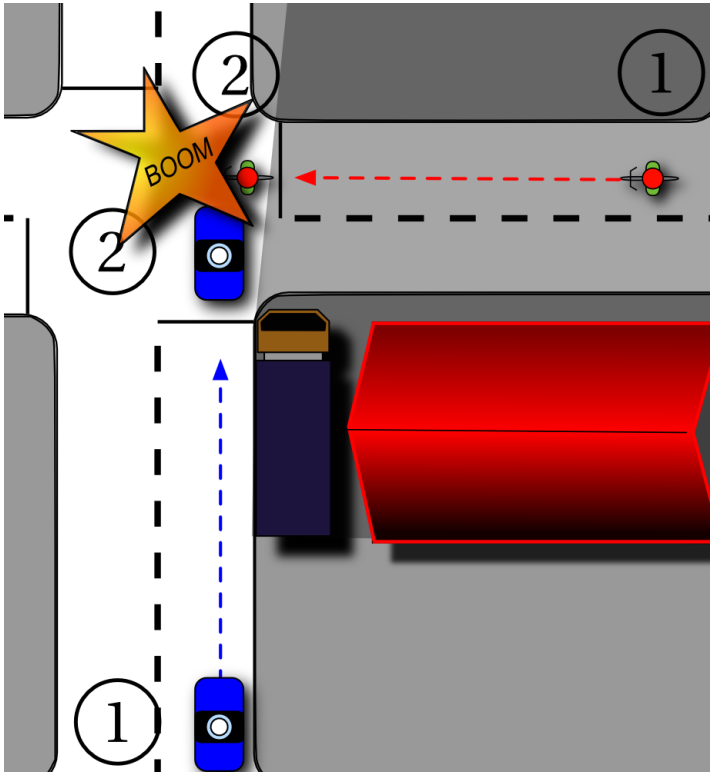
- Why anticipatory?



Symbolic Planning for Behavior Decision
Partially Known Environment

- Why bother about incomplete perception?
- Measurements are noisy
- Biggest issue: Objects are often occluded in urban environments





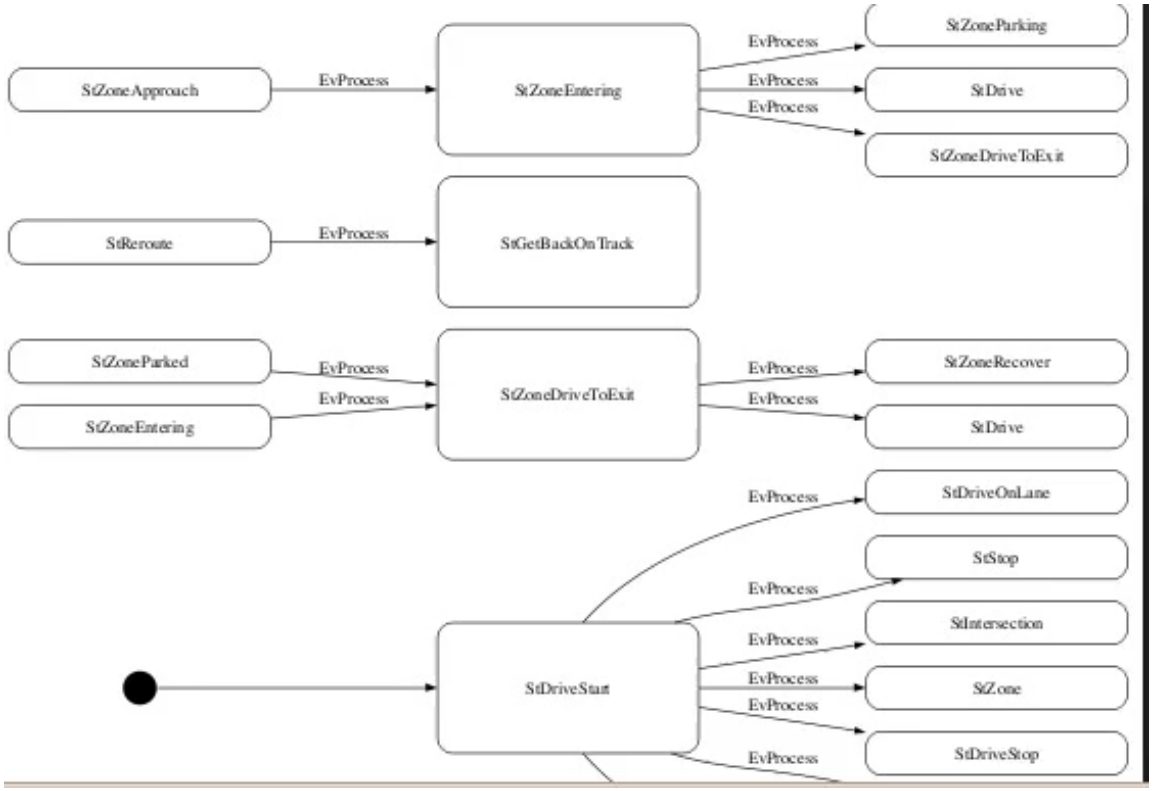
Symbolic Planning for Behavior Decision Partially Known Environment

- Beobachtungsmodell → POMDP

Symbolic Planning for Behavior Decision
Partially Known Environment

- Why not model decisions manually?
- Continuous spaces: $s, s', o \in \mathbb{R}^n$

- Task is very complex and thus error-prone:

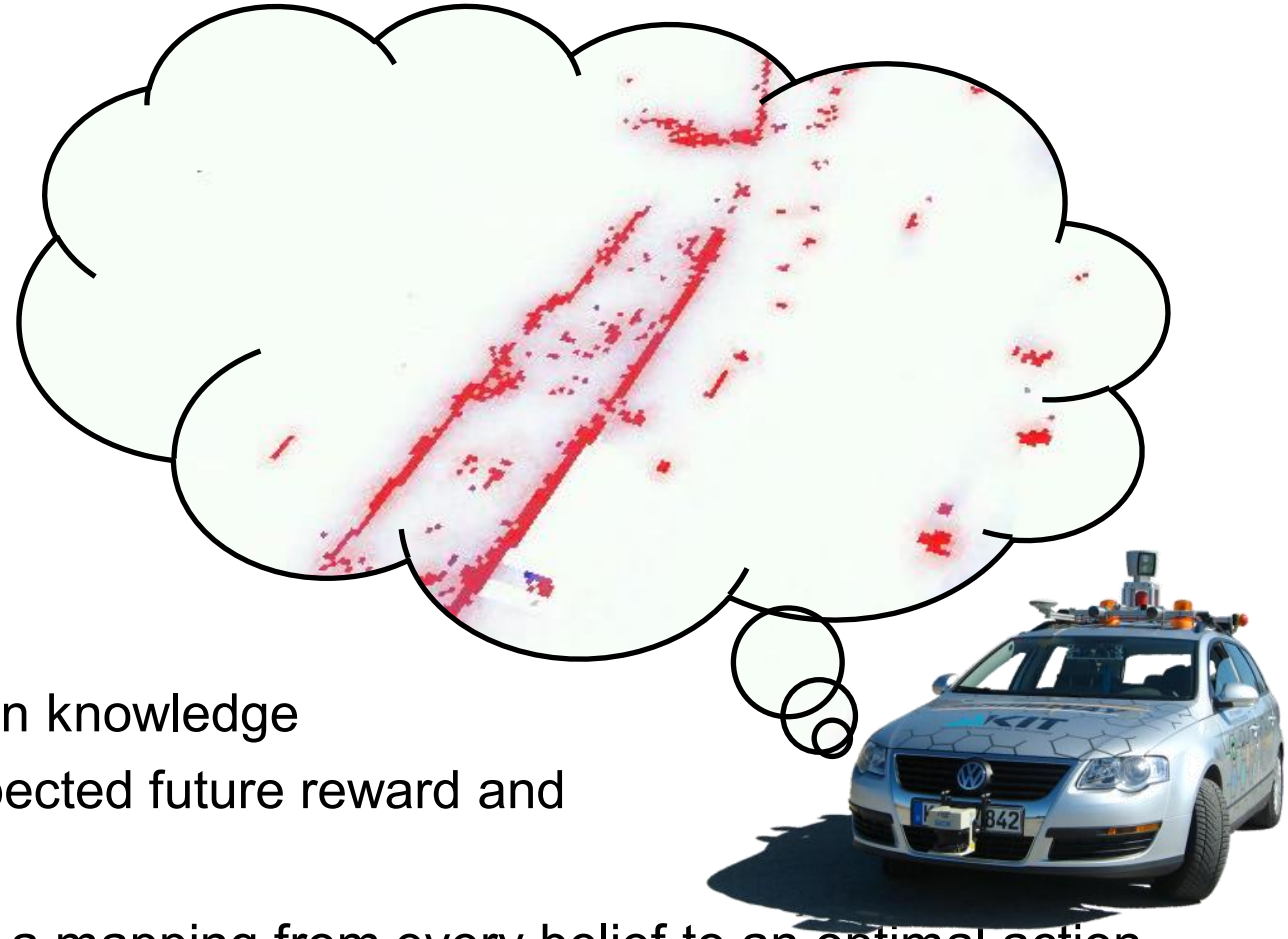


- ➔Automatic solving process necessary

AnnieWAY's Hierarchical State Machine for the Urban Challenge 2007

Symbolic Planning for Behavior Decision

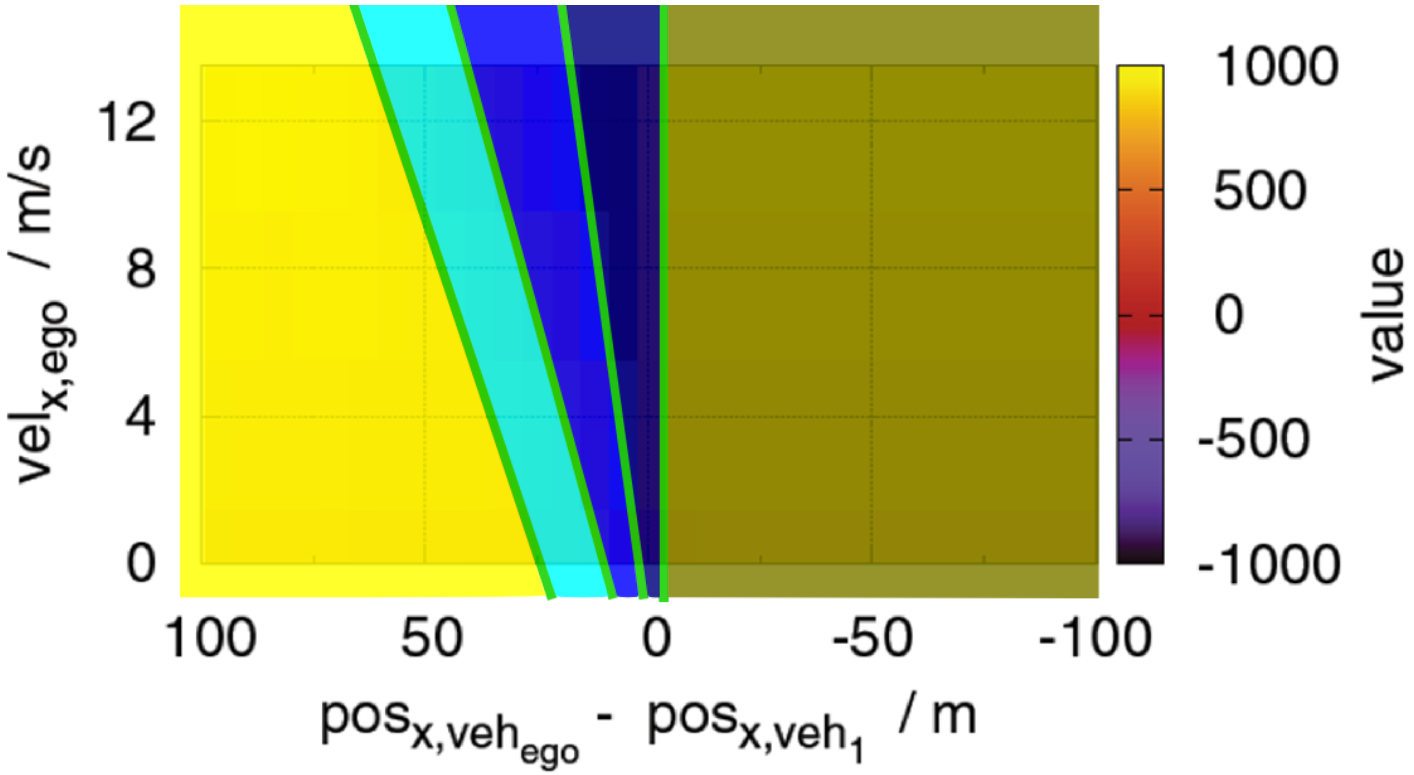
Concept



- Uncertain situation knowledge
- Maximize the expected future reward and minimize the risk
- Planning result is a mapping from every belief to an optimal action

Symbolic Planning for Behavior Decision
Relevance-based space Representation

- Equidistant discretization inefficient
  - Too fine and too coarse at the same time
  - ➔ Concept: Only consider differences relevant to the problem



Symbolic Planning for Behavior Decision

Overview on planning algorithm

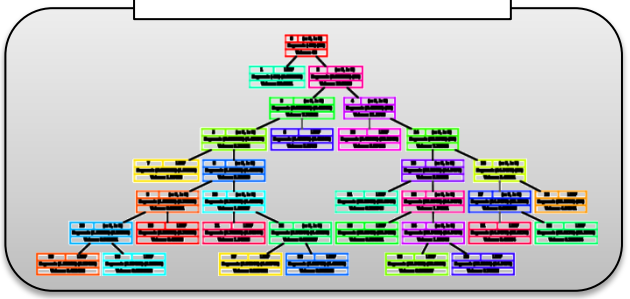
knowledge and policy base

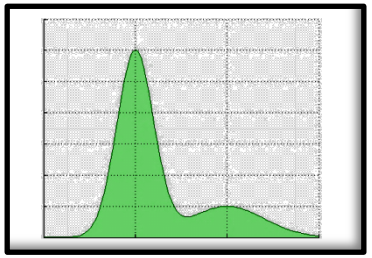
$$\alpha_{d,1} = \begin{pmatrix} \alpha_{1,1} \\ \vdots \\ \alpha_{1,n} \end{pmatrix}, \dots, \alpha_{d,k}$$

discrete belief pdf

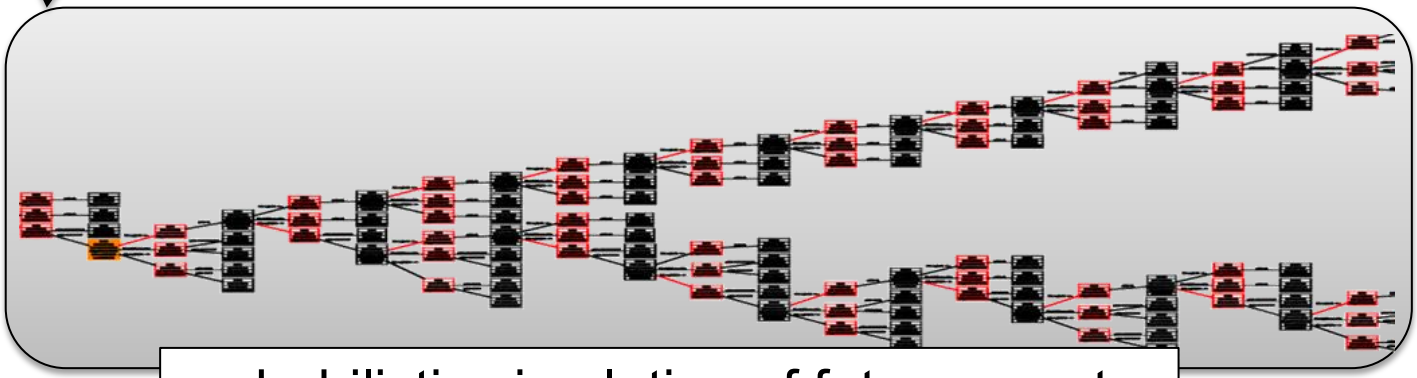
$$b_d = \begin{pmatrix} b_{1,1} \\ \vdots \\ b_n \end{pmatrix}$$

discretization





guides exploration



probabilistic simulation of future events

Symbolic Planning for Behavior Decision

MC Backup

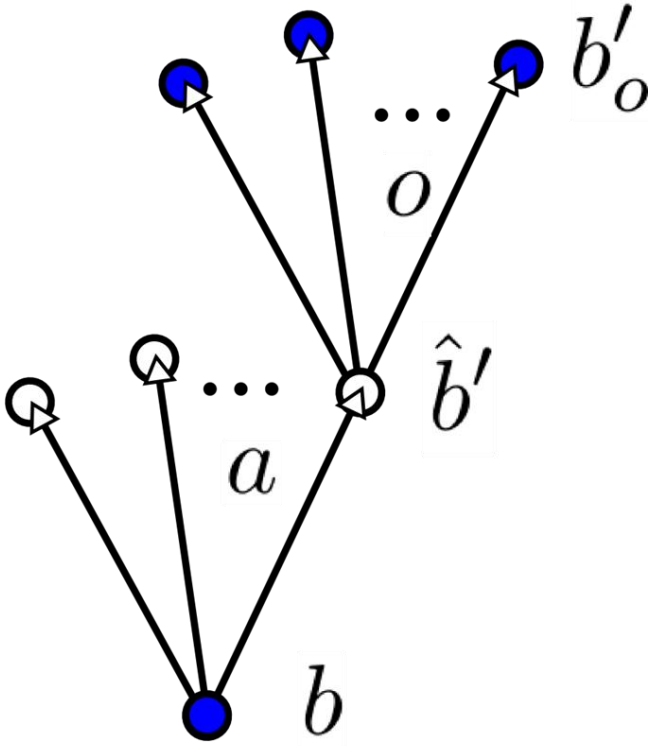
$$V'_a(b) = \underbrace{r(b,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount}} \underbrace{p(\hat{b}'|b,a)}_{\text{Prediction}} \int_o \underbrace{p(o|\hat{b}')}_{\text{Observation}} \underbrace{\alpha(b'_o)}_{\alpha\text{-Value}} \, \mathrm{d}o$$

$$s, s', o \in \mathbb{R}^n$$

$$\alpha(b') = \int_{s'} b'(s')\alpha(s') \, \mathrm{d}s' \approx$$

$$\sum_k (\alpha_d(k) b_d(k)) =< \alpha_d, b_d >_d$$

$$k \in \mathbb{N}$$

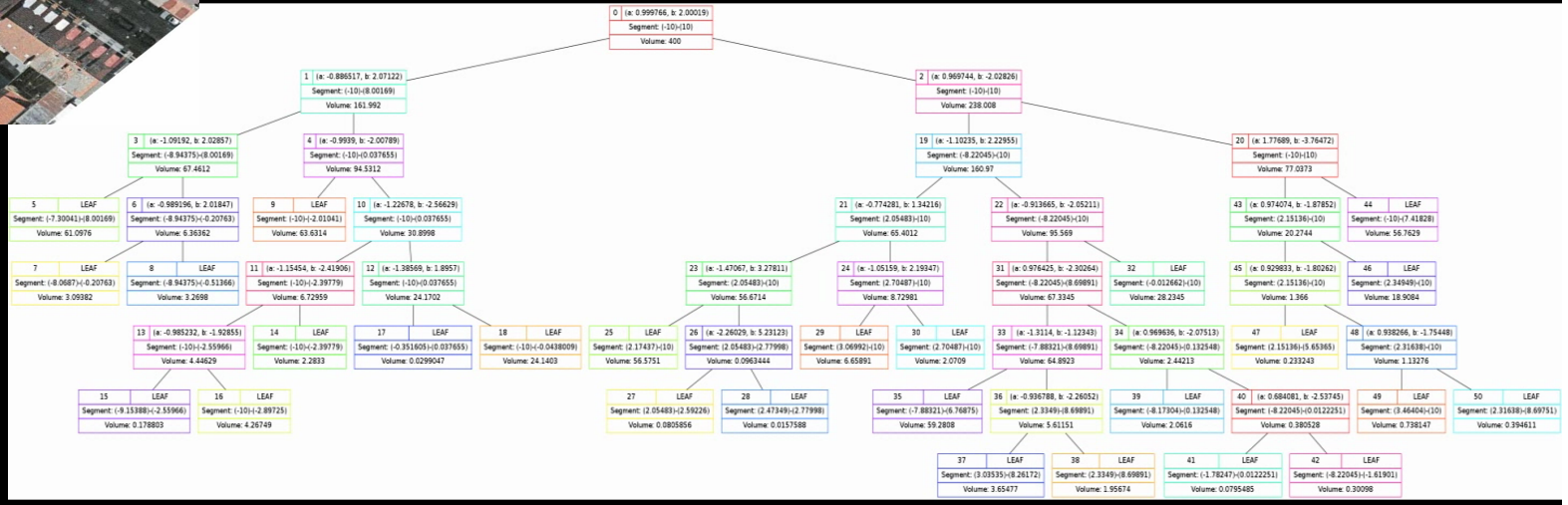


Symbolic Planning for Behavior Decision
Adaptive Refinement of Space Discretization

- If value is equal, no need to differentiate regions
- Split two regions, if the lead to different value results
- Creates

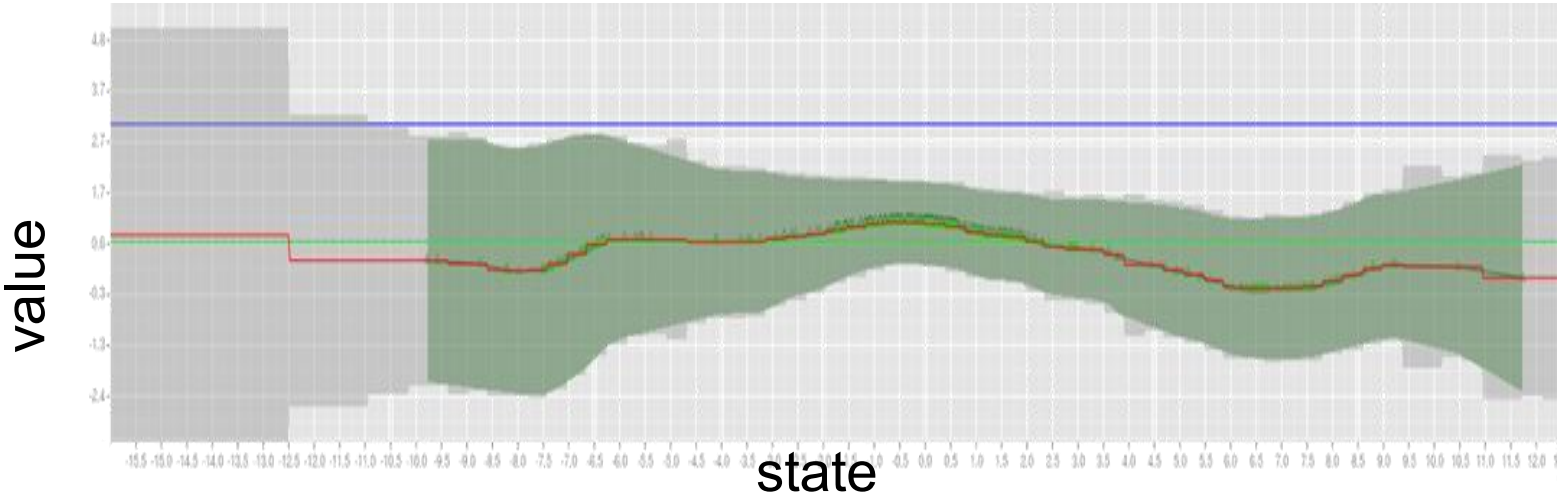
Symbolic Planning for Behavior Decision

Refining the Decision Tree Representation



Symbolic Planning for Behavior Decision
Value as Continuous α-functions

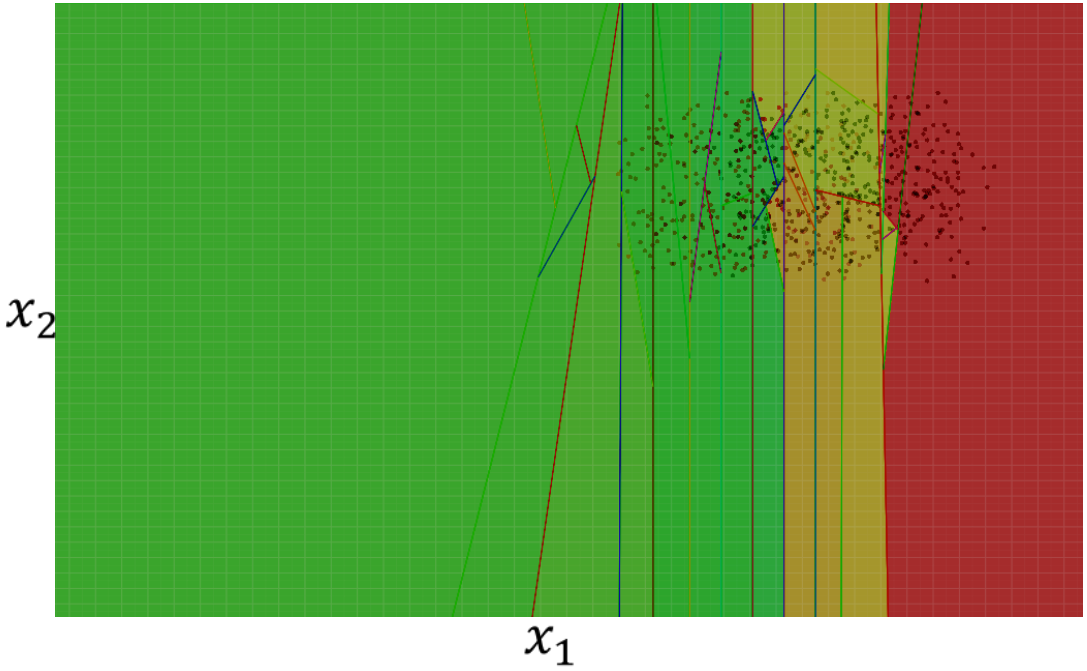
- Continuous α-function example:

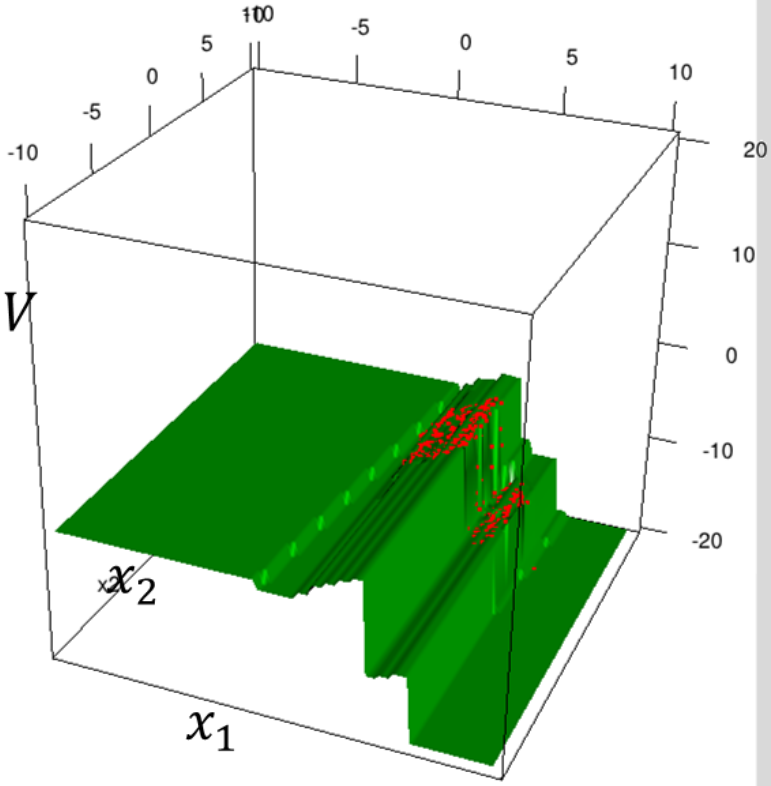


- Central discretization unit (e.g., Decision Tree) translates b to discrete space b_d in O(n log(n))
- Quick evaluation of all α_d by (sparse) dot product $< \alpha_d, b_d >_d$
- Computational complexity of dual, discrete problem

Symbolic Planning for Behavior Decision
Dimension reduction

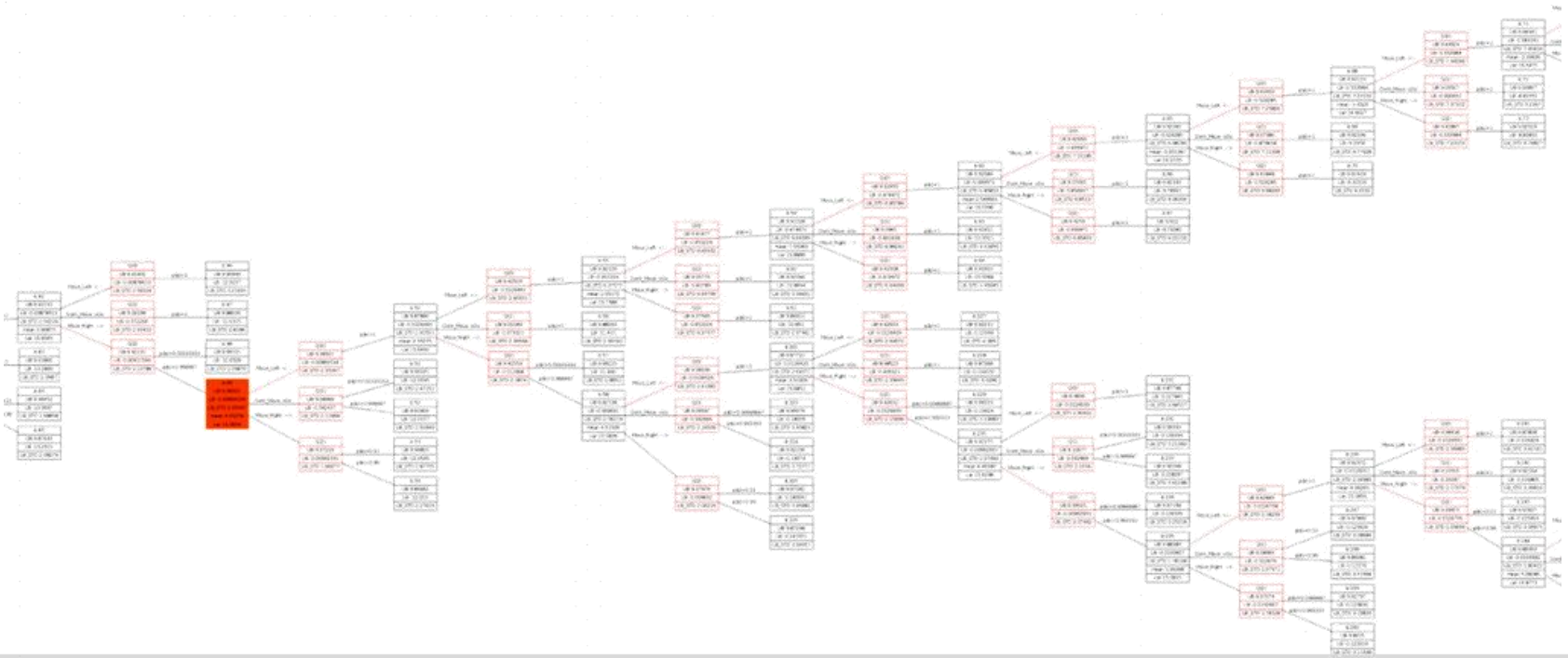
- Example: x_2-dimension meaningless for solution
- Representation focusses to describe x_1-dimension
  - ➔ Effectively reducing complexity





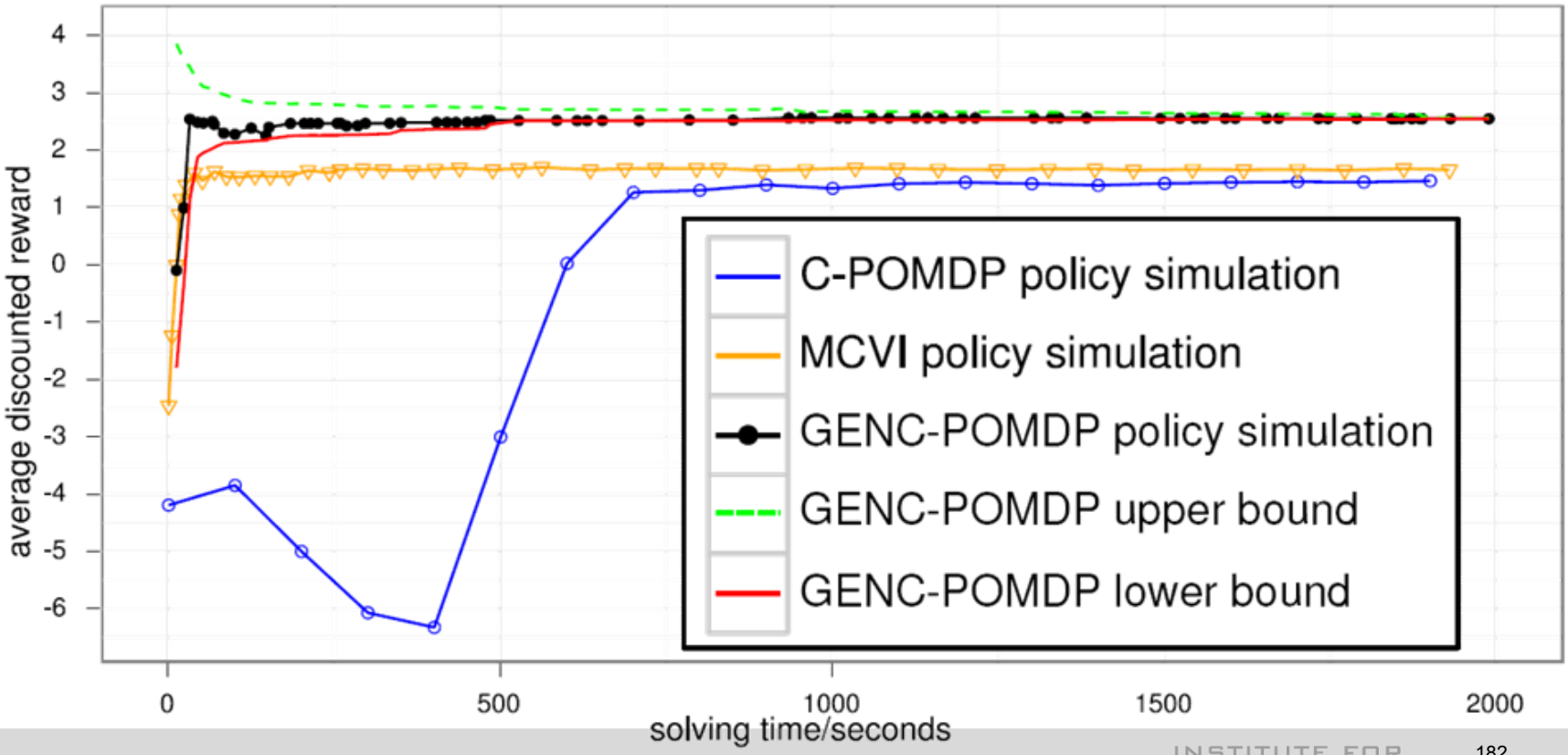
Symbolic Planning for Behavior Decision
Tree of Predicted Beliefs

- Predicts possible events in the future
- Models (prediction and observation) as arbitrary Dynamic Baysian Networks
- Close to reality by learned behavior models of other traffic participants
- MC realization realization by particle inference



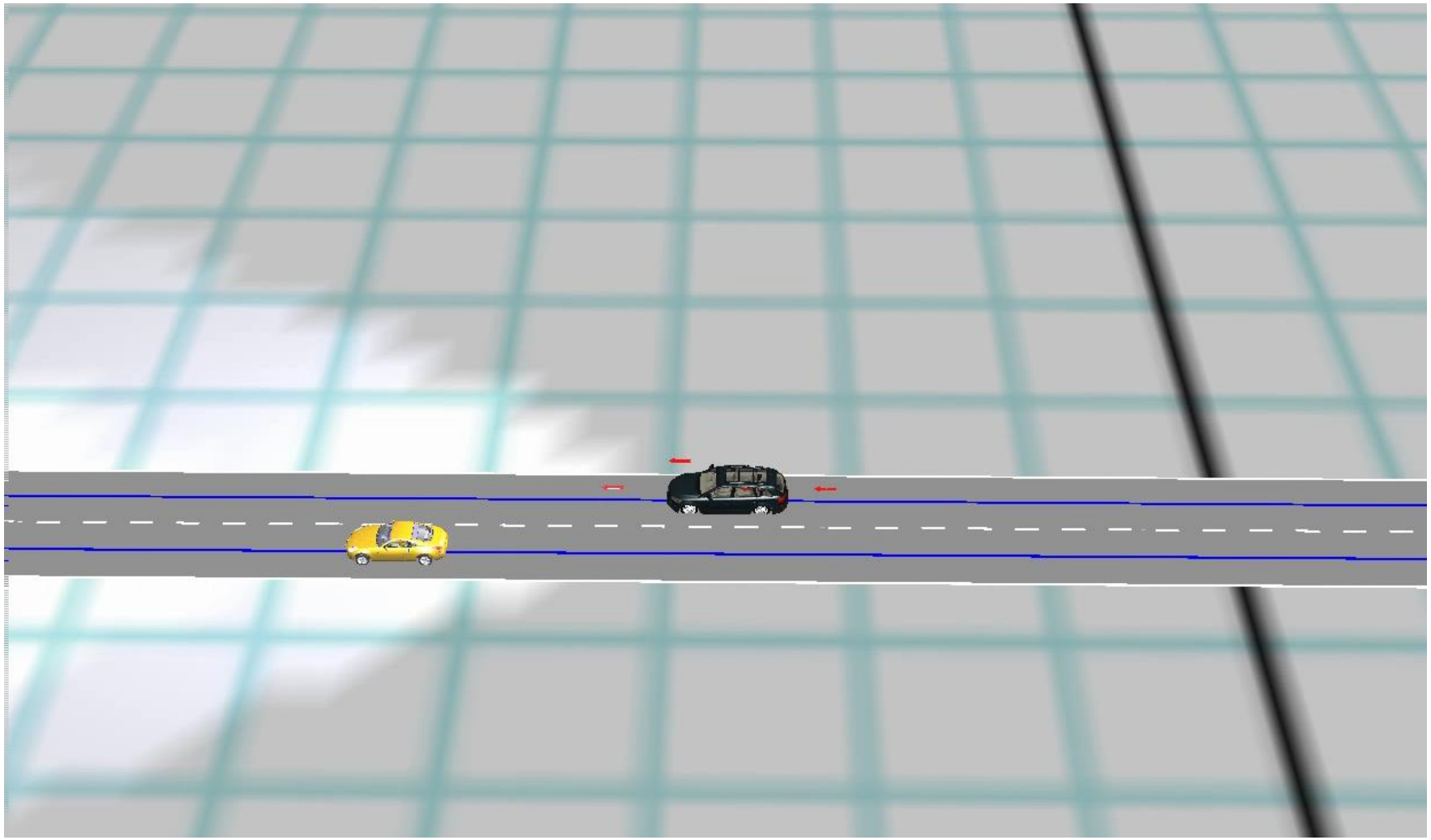
Symbolic Planning for Behavior Decision
Convergence

- GENC-POMDP creates policies with better performance in significantly less time than previous approaches
  MCVI [Bai; Hsu 11] and C-POMDP [Porta et al. 06]



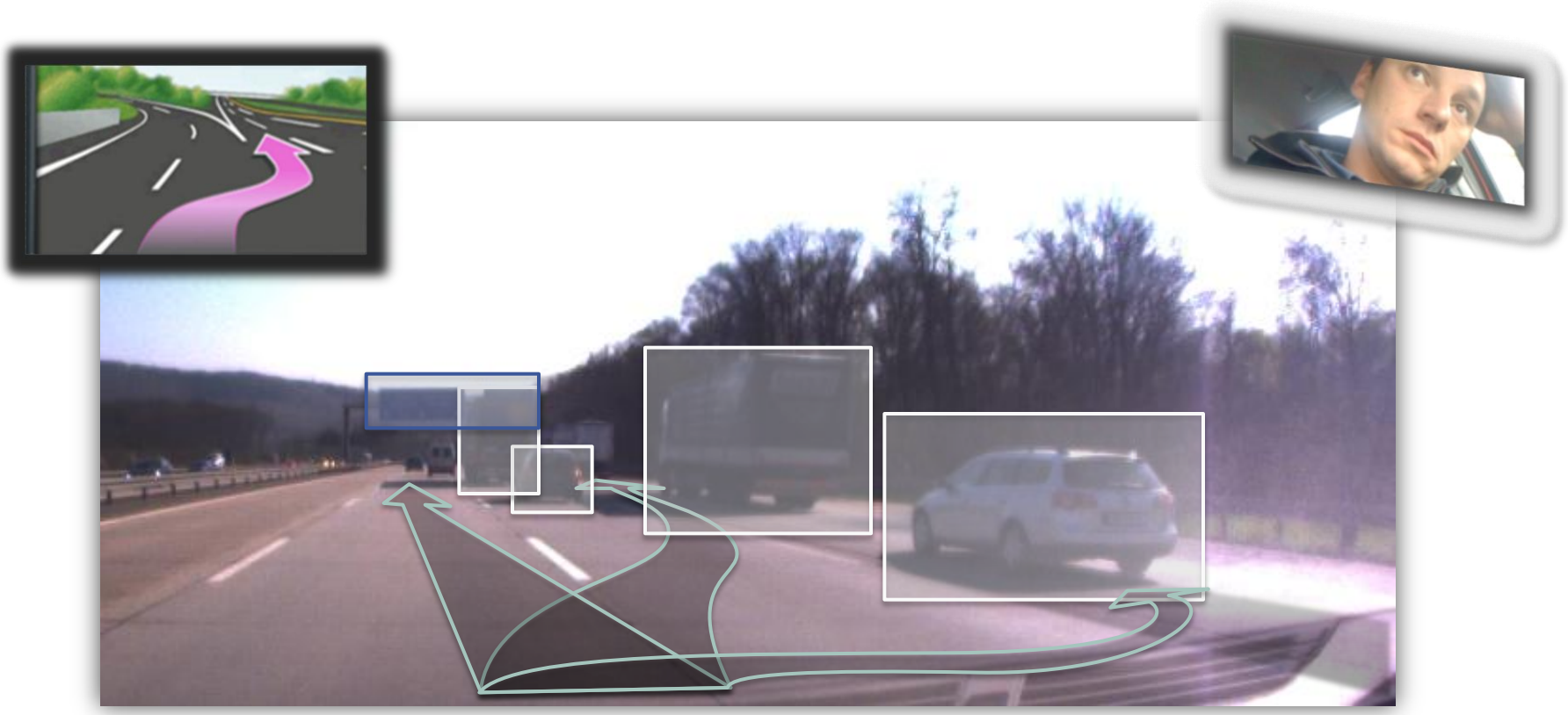
Symbolic Planning for Behavior Decision

MDP-Results



APPLICATIONS

Driver Assistance based on scene understanding and anticipation



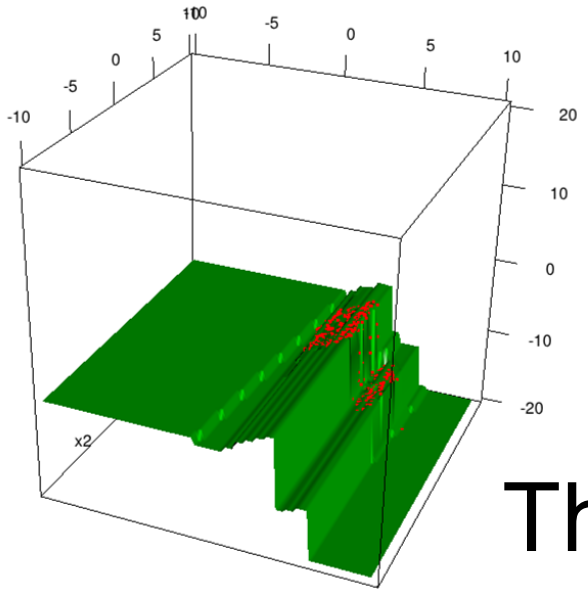
- Decide which maneuver is the safest, most efficient, etc.
- Support user to perform safe maneuver

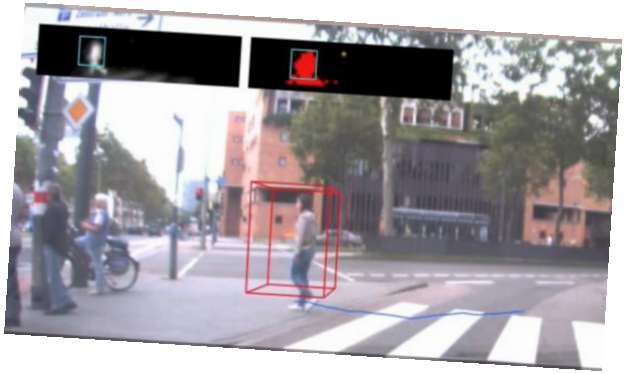
Autonomous Driving

- CoCar- The instrumented test vehicle of the FZI Living Lab Automotive

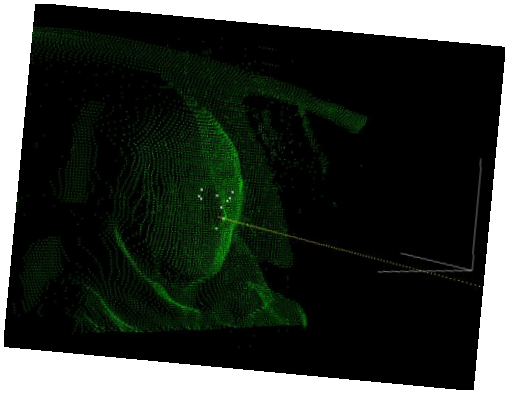
Virtual testing of autonomous parking on parking sites

- Modelling of virtual testing environment in IPG CarMaker
- Update vehicle position in CarMaker based on GPS/SLAM position estimation
- Occupied parking lots and sensor information acquired by virtual sensors
- Vehicle position estimation and path planning on virtual sensor data
- Autonomous vehicle control applyed on testing platform





Thank you for your attention…



2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session IV

Navigation, Control, Planning

- **Title: An Efficient Heuristic Estimate for Non-holonomic Motion Planning**
  Authors: J.W. Choi

- **Title: Short term path planning using a multiple hypothesis evaluation approach for an autonomous driving competition**
  Authors: M. Oliveira, V. Santos and A.D. Sappa

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

An Efficient Heuristic Estimate for Non-holonomic Motion Planning

Ji-Wung Choi

***Abstract*— A new efficient and admissible heuristic estimate function is proposed for non-holonomic motion planning. The heuristic calculation begins by partitioning a configuration space into visible and invisible spaces from the perspective of the goal configuration. The heuristic values of visible configurations are assigned by pre-computed heuristics through full state space assuming empty environment. The heuristics are extended, through the reduced Euclidean $2D$ space, into the invisible configurations by using dynamic programming. The numerical simulations demonstrate remarkable performance improvement in motion planning quries by applying the heuristic function, compared to other existing heuristics [1] and [2].**

I. INTRODUCTION

Motion planning is crucial to achieve autonomy of mobile robots. Heuristic search algorithm, such as A^*, is one of the most widely used algorithms to find the solution of motion planning problems for its performance and accuracy. The algorithm benefits from heuristic knowledge that guides search into promising directions to the goal configuration. Clearly, more accurate heuristic to estimate the true path costs leads to better performance. Conventional heuristic functions such as Euclidean distance may not be adequate for non-holonomic path search, because it underestimates path costs by violating kinematic constraints of robots. These underestimated heuristics can mislead search and thereby aggravate the bottleneck situation as the size of the search space grows. If we assume completely empty environment, feasible path costs can be pre-computed offline and stored in a Heuristic Look-Up Table (HLUT) [3]. However, the HLUT cannot consider obstacles since it is implausible to encode all possible worlds. So, a computationally efficient heuristic in large and complex environment has remained challenging.

One of the most effective heuristic functions in the literature may be the one applied for autonomous ground vehicles of CMU [1] and Stanford [2] teams in the DARPA Urban Challenge (DUC). The heuristic is given by the maximum of two component heuristics: 1) non-holonomic-without-obstacle (stored in HLUT) and 2) holonomic-with-obstacle. While the heuristic is well informed in local area around the goal configuration, it often underestimates the actual path costs in other area (as more detailed in Section II-B).

This motivated us to propose a novel, hybrid heuristic estimate function. The first step of computing the heuristic is to divide the state space into two: 1) visible and 2) invisible spaces from the goal state's perspective. Heuristic values of the visible space are simply copied from HLUT. For the invisible space, though, complexity is inherited from existence of obstacles. We cope with this complexity by projecting the visible heuristic values into the reduced Euclidean $2D$ space and applying dynamic programming to extend the values into the invisible space. As the result, the heuristic becomes admissible and consistent. The simulation results provided in Section IV demonstrate benefits from the new heuristic function in motion planning query.

This work was supported by the Academy of Finland under GIM project.

J.-W. Choi is with Department of Intelligent Hydraulics and Automation, Tampere University of Technology, 33101 Tampere, Finland ji.choi@tut.fi

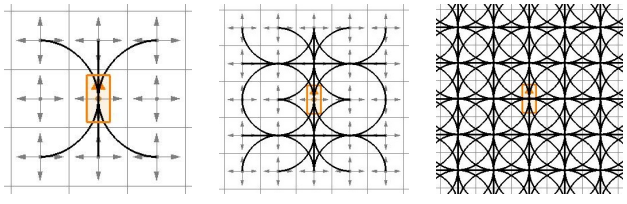
II. PRIOR WORK

A. State Lattice

Many work on search algorithms provide computationally efficient way in discrete state spaces [4], [5]. However, the resulting paths by such algorithms do not tend to be smooth (piecewise linear) and hence, do not satisfy kinematic feasibility of the robot. In order to satisfy motion constraints while achieving the computational advantages of discretization, Pivtoraiko and Kelly proposes the *state lattice* [6]. The state lattice is a graph representation of a discretized configuration space, where repeating pairs of nodes (configurations or states) are connected by a finite set of feasible path segments, referred to the *control set*. Each control must be subject to robotic systems' dynamics such as non-holonomic nature and the maximum curvature constraint so that the constrained motion planning is formulated into query as graph search.

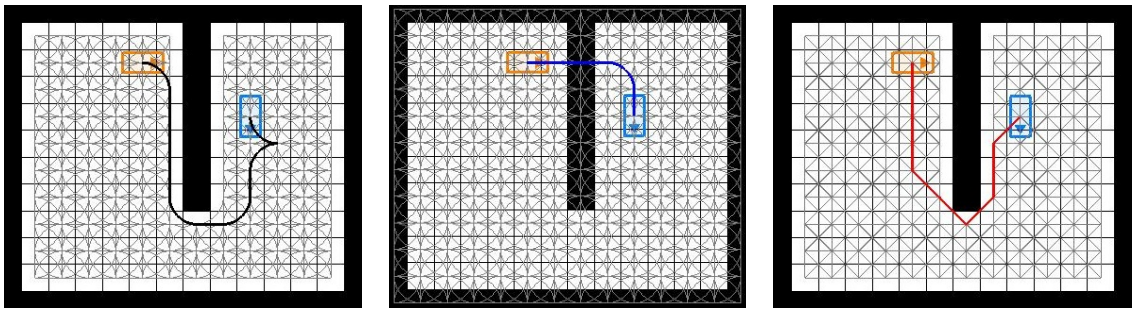
Fig. 1 illustrates an example of $3D$ state lattice (x, y, θ). A search graph is constructed by copying the control set at states in a configuration space, $\mathcal{C}_{space}$ (Fig. 1(c)). The advantage of the lattice planner comes from kinematic feasibility requirement of lattice connections. As such, a sequence of controls in the graph forms a path that guarantees feasibility.

Due to the compactness, we describe our new heuristic function in the state lattice frame depicted in Fig. 1.



(a) (b) (c)

Fig. 1. A $3D$ state lattice construction. (a) The lattice is composed of discretized position (x, y) and headings θ, illustrated as dots and arrows, respectively. The control set (thick solid curves) is a set of elementary feasible motions to connect each state and its reachable neighbors. (b) The reachability tree obtained by proceeding 2 steps with the control set. (c) The complete reachability tree is constructed by copying the control set at states in a $\mathcal{C}_{space}$.



(a) Nonholonomic-with-obstacle for h^{+}_{3D} (b) Nonholonomic-without-obstacle for h^{-}_{3D} (c) Holonomic-with-obstacle for h^{+}_{2D}

Fig. 2. Search graphs representation. (a) The nonholonomic-with-obstacle graph is constructed by copying the control set elements that do not hit $\mathcal{C}_{obstacle}$ to the $3D$ (full dimension) states in free space. Since each control is generated subject to kinematic constraints, a sequence of controls in the graph forms a safe path that guarantees feasibility. (b) The nonholonomic-without-obstacle graph is constructed by copying the control set at every state in a $\mathcal{C}_{space}$, without considering obstacles. While paths in the graph guarantee feasibility, they can collide with obstacles. (c) The holonomic-with-obstacl graph consists of 8-(or 4-)connected $2D$ states (positions) in free space. Paths in the graph guarantee obstacle avoidance but violate kinematic constraints; Given the initial and goal state (red and blue rectangle), costs of the optimal solutions (thick solid curves) through the graphs defined in (a), (b), and (c) are denoted h^{+}_{3D}, h^{-}_{3D}, and h^{+}_{2D}, respectively.

B. Maximum Heuristic

The heuristic search algorithms have been widely used to achieve autonomy of mobile robots operating in dynamically changing environment. A heuristic is intended to improve search performance by guiding the search in promising directions. The performance significantly depends on the accuracy of heuristic estimation for actual path cost. The accuracy comes at a price of computational complexity. Although, for example, naive $2D$ Euclidean distance heuristic is computationally cheap, it often underestimates path costs by violating non-holonomic nature of the robot. Conversely, the optimal solution in the $3D$ (full dimension) state lattice embedded in a free space, as in Fig. 2(a), may perfectly estimate the costs under the limit of the lattice resolution. (Let us denote the cost function h^{+}_{3D}.) However, it is computationally impractical as the search space grows.

So, it still has remained challenging to provide a heuristic function that balances estimate accuracy and computational efficiency. This subsection introduces one of the most effective approaches in the literature. The heuristic has been applied for unmanned ground vehicles of the CMU [1] and Stanford [2] teams in the DARPA Urban Challenge. We call the heuristic the *maximum heuristic* because it is given by the maximum of two component heuristics: 1) non-holonomic-without-obstacle and 2) holonomic-with-obstacle.

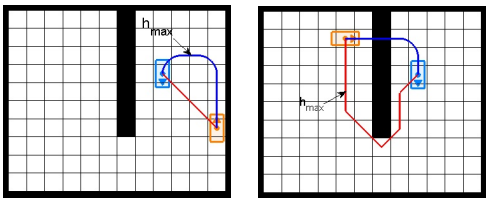
The non-holonomic-without-obstacle heuristic, denoted h^{-}_{3D}, is the cost of an optimal solution in the $3D$ (full dimension) state lattice copied at a $\mathcal{C}_{space}$ without considering obstacles. Fig. 2(b) shows the search graph in which h^{-}_{3D} is defined. The heuristic can be pre-computed and stored as a Heuristic Look-Up Table (HLUT). Then, it is translated and rotated with respect to the goal state. While the heuristic is well informed in sparse spaces, it gradually underestimates the true costs in dense spaces due to the lack of obstacle information. (Compare the solution in Fig. 2(b) with that in Fig. 2(a).) The reason why obstacles can not be taken into account on HLUT is because there are nearly infinite number of possibilities for obstacle-laden environments.

The holonomic-with-obstacle heuristic, denoted h^{+}_{2D}, copes with the complexity inherited from existence of obstacles. The cost is obtained by running Dijkstra's algorithm on the reduced $2D$ Euclidean state space with considering obstacles. Fig. 2(c) shows the search graph in which h^{+}_{2D} is defined. While the heuristic can be calculated online by virtue of state dimension reduction, it may underestimate the true path costs by ignoring non-holonomic nature. (Compare the solution in Fig. 2(c) with that in Fig. 2(a).)

The maximum heuristic, denoted h_{max}, takes the maximum of the two heuristics:

$$h_{max}(x,y,\theta) = \max(h^{-}_{3D}(x,y,\theta), h^{+}_{2D}(x,y)),$$

and thereby gains an order of magnitude performance improvement than either of the two component heuristics [1].

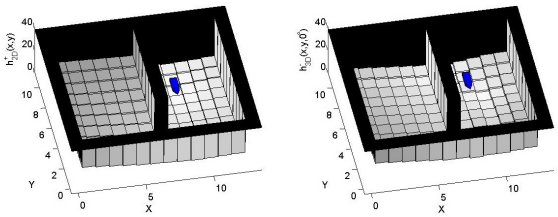


(a) $h^{-}_{3D} > h^{+}_{2D} \Rightarrow h_{max} = h^{-}_{3D}$ (b) $h^{+}_{2D} > h^{-}_{3D} \Rightarrow h_{max} = h^{+}_{2D}$

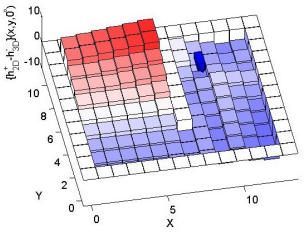
Fig. 3. The maximum heuristic value of a state (x,y,θ) (red rectangle) is determined by $h^{-}_{3D}(x,y,\theta)$ or $h^{+}_{2D}(x,y)$, depending on the location of (x,y) relative to the goal (blue rectangle). Blue and red curves represent solutions of h^{-}_{3D} and h^{+}_{2D}, respectively. (a) If (x,y) lies in open local area around the goal, h_{max} is likely to be determined by h^{-}_{3D}. (b) If an obstacle lies between (x,y) and the goal, h_{max} is likely to be determined by h^{+}_{2D}.

Fig. 3 shows that $h_{max}(x,y,\theta)$ is determined by $h^{-}_{3D}(x,y,\theta)$ or $h^{+}_{2D}(x,y)$ depending on the relative location of (x,y) to the goal state. Referring to Fig. 3(a), if a state (red rectangle) lies in open local area around the goal state (blue rectangle), then h_{max} is likely to be determined by h^{-}_{3D} (blue curve). Otherwise, as shown in Fig. 3(b), h_{max} is likely to be determined by h^{+}_{2D} (red linear path).

This position dependency of h_{max} is clearly visualized in Fig. 4(c). The figure shows the differences between the two



(a) $h^+_{2D}(x,y)$ (b) $h^-_{3D}(x,y,0^\circ)$



(c) $h^+_{2D}(x,y) - h^-_{3D}(x,y,0^\circ)$

Fig. 4. Given the environment in Fig. 3, (a) $h^+_{2D}(x,y)$ and (b) $h^-_{3D}(x,y)$ with $\theta = 0^\circ$ are displayed. (c) The differences $h^+_{2D}(x,y) - h^-_{3D}(x,y,0^\circ)$ represent position dependency of $h_{max}(x,y,0^\circ)$: h_{max} is determined by h^+_{2D} (or h^-_{3D}) at the positive (or negative) valued positions filled with red (or blue).

heuristics, $h^+_{2D}(x,y) - h^-_{3D}(x,y,0^\circ)$, for the $\mathcal{C}_{space}$ and the goal state shown in Fig. 3. While negative values (blue), in which h_{max} is determined by h^-_{3D}, are located in the relatively open space close to the goal, positive values (red), in which h_{max} is determined by h^+_{2D}, are located in the space concealed behind $\mathcal{C}_{obstacle}$ from the perspective of the goal.

The heuristic determined by h^-_{3D} in the blue area relatively well estimates true optimal path costs. Although, on the other hand, h^+_{2D} estimates better than h^-_{3D} in the red area, it still underestimates true path costs. For example, let us compare the red linear path in Fig. 3(b) with the optimal solution in Fig. 2(a). While the latter involves the robot making an U-turn to reach the goal state, the first simply consists of line segments, and hence informs shorter distance (smaller cost) than the latter. The underestimated heuristic can mislead the search into the exploration of wrong regions. This motivated us to propose a novel, hybrid heuristic estimate function.

III. The Proposed Hybrid Heuristic

This section proposes a new heuristic estimate function to overcome the drawback of the maximum heuristic, described above. The proposed heuristic has been created based on the following observations:

- $2D$ configuration space is relatively well divided into two separate regions, depending on the comparison between h^-_{3D} and h^+_{2D} (as in Fig. 4(c)).
- In the region closer to the goal (so that the goal is visible), h^-_{3D} well estimates optimal costs.
- In the other region (where the goal is invisible), h^+_{2D} estimates better than h^-_{3D}, but underestimates true optimal costs.

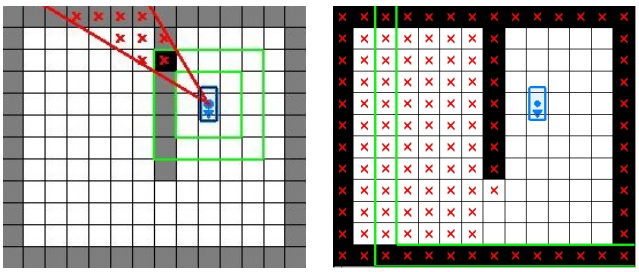
From the observations above, we introduce the concept of the *visibility* (from the perspective of the goal position) to divide $2D$ configuration space into two: visible and invisible spaces (detailed in Section III-A). Since h^-_{3D} performs well in the visible space, its heuristic is simply copied from HLUT into the region. Then, $2D$ version costs are extended from the visible to invisible space by using dynamic programming. As the result, the heuristic of the invisible space is a combination of non-holonomic and holonomic path costs and hence estimates better than pure h^+_{2D} (detailed in Section III-B).

A. Visible Space

The visible space $\mathcal{V} \subset \mathbb{R}^2$ is the set of the positions from which the goal (x_f, y_f) is visible. More specifically, if the line connecting (x,y) and (x_f, y_f) intersects no obstacle, then (x,y) is defined to be visible, otherwise invisible.

Algorithm 1 Visible Space Construction

```
1:  procedure BUILDVSPACE(x_f, y_f, C_space)
2:      Mark all (x,y) ∈ C_space as visible.
3:      [x_min, x_max, y_min, y_max] ⇐ size(C_space)
4:      L_max ⇐ max(x_max − x_f, x_f − x_min, y_max − y_f, y_f − y_min)
5:      for i = 1 → L_max do
6:          for all (x,y) s.t. L_∞([x,y] − [x_f, y_f]) = i do
7:              if (x,y) ∈ C_obstacle then
8:                  [x', y'] ⇐ InvisibleStates(x, y, x_f, y_f)
9:                  Mark all (x', y') invisible.
10:             end if
11:         end for
12:         if all (x,y) are marked as invisible then
13:             break
14:         end if
15:     end for
16:     return V ⇐ {(x,y) | (x,y) is marked as visible}
17: end procedure
```



(a) Invisible states by a cell (b) Overall invisible states

Fig. 5. (a) Given the goal state (blue rectangle) and a state in $\mathcal{C}_{obstacle}$ (black cell), invisible states (red crosses) concealed behind the cell lies inside of two tangential rays (red lines) from the goal to the cell. (b) The invisible space (filled with red crosses) is the set of all invisible states by $\mathcal{C}_{obstacle}$. The visible space (white cells) is the difference set of the invisible space.

Algorithm 1 is the pseudo code for constructing the visible space $\mathcal{V}$. Initially, all $2D$ states in a $\mathcal{C}_{space}$ are marked as *visible* (Line 2). Then, it iterates exploring (x,y) on the square boundary around (x_f, y_f) as its maximum norm[1] L_∞ grows from 1 until it reaches the boundary of $\mathcal{C}_{space}$ (Line 6 -11). If (x,y) is in $\mathcal{C}_{obstacle}$ (Line 7), then all the states

[1] $L_\infty([x,y]) = \max(|x|, |y|)$

concealed behind (x,y) from the perspective of (x_f,y_f) are calculated (Line 8) and marked as invisible (Line 9). Fig. 5(a) visualizes the invisible states (red crosses) by an obscured cell (black). The invisible states lie inside of two tangential rays from (x_f,y_f) to the cell. They can be pre-computed and stored in a look-up table to speed up the construction. The procedure terminates as all the (x,y) on the square boundary (green lines) is invisible as shown in Fig. 5(b) (Line 12).

B. The Hybrid Heuristic Construction

The hybrid heuristic is calculated based on the visible space, as presented in Algorithm 2. In the pseudo-code, Θ is the canonical set of discrete headings, and ρ is the threshold distance to compensate underestimate effect by holonomic-with-obstacle path costs (more detailed in next subsection). The algorithm is mainly divided into two loops: one cycles through the visible space (Line 5-16) and the other through the invisible space (Line 17-20). The first loop copies $h^-_{3D}(x,y,\theta)$ to all visible states (Line 7). As stated in Section II-B, the heuristic values are pre-computed offline and stored in HLUT so that the copy performs in constant time complexity. The copied heuristics are extended to heuristics of invisible states. Since invisible heuristics are defined in $2D$, we project the visible heuristic values $h^-_{3D}(x,y,\theta)$ onto $2D$ version, $H_{2D}(x,y)$ with the minimum for all $\theta \in \Theta$ (Line 9). If a visible $2D$ state (x,y) has an invisible successor (x',y'), then the heuristic of the successor is updated with the minimum of sum of $H_{2D}(x,y)$, $\|(x,y)-(x',y')\|$, and ρ (Line 12-13). Once all visible states has been copied, H_{2D} will have contained heuristics for all the visible states and the invisible states adjacent to the visible space. Finally, we run dynamic programming in $2D$ space to solve the shortest distance from each invisible state to the goal with incorporating the existing H_{2D} (Line 18).

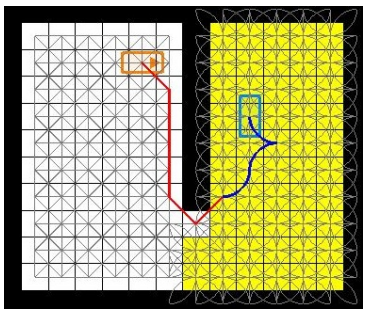


Fig. 6. The hybrid heuristic graph is constructed by copying h^-_{3D} graph at visible space (yellow cells) and h^+_{2D} graph at invisible space (white cells). Thus, the optimal solutions through the hybrid graph combines h^-_{3D} (blue) and h^+_{2D} paths (red).

Fig. 6 shows the hybrid heuristic graph and a solution through the graph. In the visible space filled with yellow, h_{hyb} is determined by non-holonmic-without-obstacle path cost as h_{max} is. The difference between h_{hyb} and h_{max} is that h_{hyb} combines it with holonomic-with-obstacle path cost in the invisible space. In other words, the reference path for h_{hyb} is the combination of a holonomic path (the red linear path)

Algorithm 2 Hybrid Heuristic Estimate

```
1:  procedure HYBRIDHEURISTIC(V, C_space, Θ, ρ)
2:      for all (x,y) ∈ C_space do
3:          H_2D(x,y) ⇐ ∞
4:      end for
5:      for all (x,y) ∈ V do
6:          for all θ ∈ Θ do
7:              h_hyb(x,y,θ) ⇐ h^-_3D(x,y,θ)
8:          end for
9:          H_2D(x,y) ⇐ min_{θ∈Θ} h^-_3D(x,y,θ)
10:         for all (x',y') ∈ Succ(x,y) do
11:             if (x',y') ∉ V then
12:                 H_t ⇐ H_2D(x,y) + ||(x,y) − (x',y')|| + ρ
13:                 H_2D(x',y') ⇐ min(H_2D(x',y'), H_t)
14:             end if
15:         end for
16:     end for
17:     for all (x,y) ∉ V do
18:         H_2D(x,y) ⇐ DynamicProgramming(x,y,H_2D)
19:         h_hyb(x,y) ⇐ H_2D(x,y)
20:     end for
21: end procedure
```

in the invisible space and a non-holonomic path (the blue curve) in the visible space as shown in Fig. 6. This hybrid path cost alleviates the underestimate of a pure holonomic path of h_{max} for invisible states. Note that while the reference path in Fig. 2(c) simply consists of line segments, the path in Fig. 6 involves making the robot U-turn to reach the goal as that in Fig. 2(a) does, As the result, h_{hyb} matches closer to the optimal cost than h_{max} does.

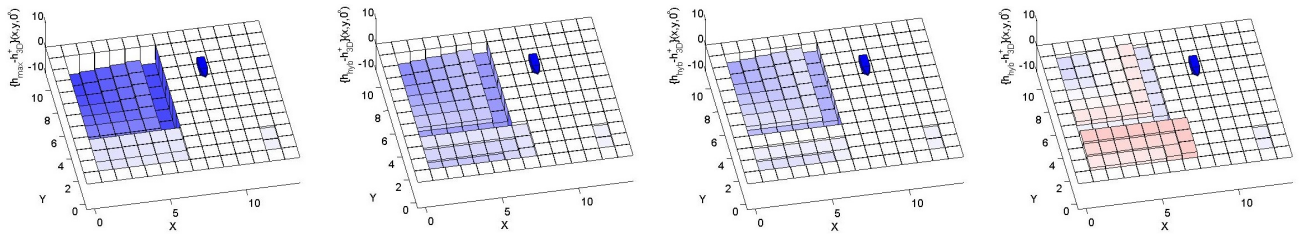
Properties of the hybrid heuristic can be summarized as follows:

- Since both h^-_{3D} and h^+_{2D} are admissible and consistent, the combined heuristic h_{hyb} is also admissible and consistent.
- h_{hyb} is constructed in same time complexity as h_{max} is. Note that h_{max} runs dynamic programming from the goal to all $2D$ states. Similarly, h_{hyb} runs dynamic programming for $2D$ states, among of which costs of visible states are copied from HLUT in constant time.
- h_{hyb} can be beneficial by saving memory for HLUT. Unlike h_{max} using HLUT for whole configuration space, h_{hyb} uses HLUT only for the visible space.

C. Threshold Distance

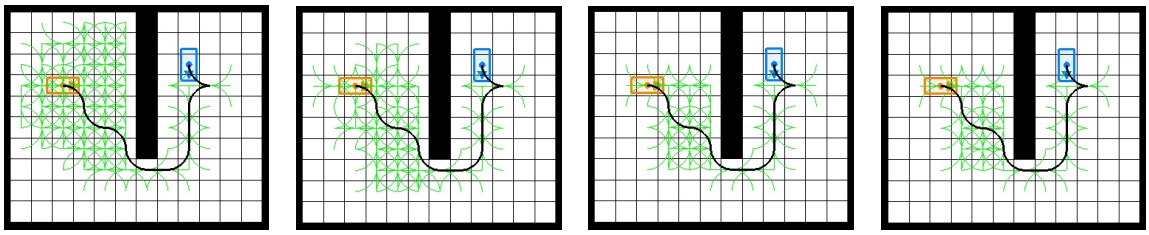
The threshold distance ρ controls the gap of heuristic values between visible and invisible spaces (Line 12 of Algorithm 2), and thereby compensates underestimating effect by holonomic paths in invisible space.

Fig. 7 shows the accuracies of h_{max} and h_{hyb} (with different ρ) to estimate h^+_{3D}, for the $\mathcal{C}_{space}$ and the goal depicted in Fig. 6. Recall that h^+_{3D} are the costs of the shortest paths through the non-holonomic-with-obstacle graph as in Fig. 2(a), and thereby perfectly estimate true optimal costs under the limit of the lattice resolution. While the graph in



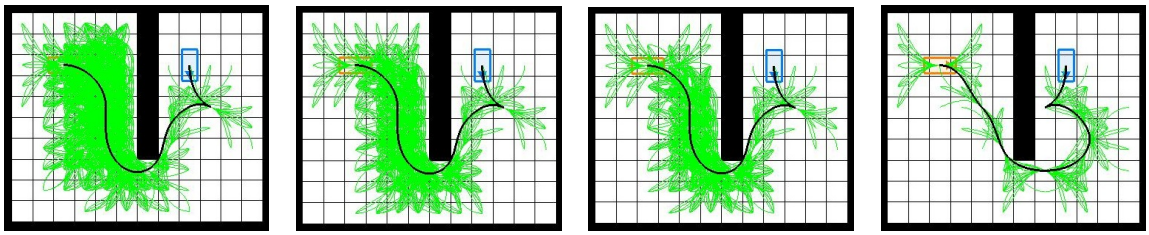
(a) $\{h_{max} - h^+_{3D}\}(x,y,0^\circ)$ (b) $\{h_{hyb} - h^+_{3D}\}(x,y,0^\circ)$ with $\rho = 0$ (c) $\{h_{hyb} - h^+_{3D}\}(x,y,0^\circ)$ with $\rho = 1$ (d) $\{h_{hyb} - h^+_{3D}\}(x,y,0^\circ)$ with $\rho = 3$

Fig. 7. Accuracies of h_{max} and h_{hyb}, to estimate optimal costs h^+_{3D}. Blueness (or redness) indicates underestimation (or overestimation).



(a) h_{max} (b) h_{hyb}, $\rho = 0$ (c) h_{hyb}, $\rho = 1$ (d) h_{hyb}, $\rho = 3$

Fig. 8. The A^* resulting paths (black curves) depending on applied heuristics. The green curves represent expanded controls.



(a) h_{max} (b) h_{hyb}, $\rho = 0$ (c) h_{hyb}, $\rho = 1$ (d) h_{hyb}, $\rho = 3$

Fig. 9. The resulting paths with the use of a higher resolution control set.

Fig. 7(a) represents $h_{max}(x,y,0^\circ) - h^+_{3D}(x,y,0^\circ)$, the graphs in Fig. 7(b)-7(d) represent $h_{hyb}(x,y,0^\circ) - h^+_{3D}(x,y,0^\circ)$ with ρ varying from 0 to 3. So, while negative values (blue) indicate underestimation, positive values (red) indicate overestimation. Zero values (white) indicate perfect estimation.

Referring to Fig. 7(a), while h_{max} nearly matches h^+_{3D} around the goal (blue cone), it underestimates in the area concealed behind the wall obstacle. This underestimated heuristics can lead to wasteful exploration of dead-ends. Applying h_{hyb} alleviates the underestimate as shown in Fig. 7(b). In addition, we can see that the underestimated heuristic grows to match h^+_{3D} as ρ grows, shown in Fig. 7(c)-7(d). At the same time, it causes overestimate in the lower left area. This may keep search from rolling backward at the area so as to speed up the search but to produce less optimal paths.

Fig. 8 shows the resulting paths by applying A^* with h_{max} and h_{hyb} with $\rho = 0, 1$, and 3. As expected, the h_{max} path undergoes wasteful exploration at the upper left area. On the other hand, h_{hyb} generates paths with less nodes expanded by virtue of better accuracies shown in Fig. 7. Fig. 9 shows the results with a higher resolution control set. We can see that the resulting path is generated with less expanded nodes but provides less optimality as ρ grows (Fig. 9(d)).

IV. Simulation

In this section we demonstrate search performance improvement by applying our new heuristic function compared to the maximum heuristic function. Simulations provided in this section were implemented in MATLAB on Intel Core i5 CPU 2.5 GHz and 4 GB RAM. The motion planner uses the control set with 16 discrete headings (as shown in Fig. 10) to generate a path.

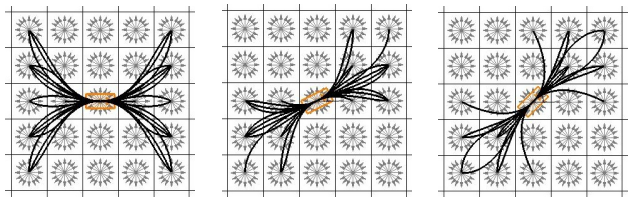


Fig. 10. The control sets allowed for the states with 0, $26.6^\circ = \arctan(1/2)$, and 45°. Reflecting the sets around $X-$ and $Y-$axes comprises overall control set with 16 total discrete headings.

Fig . 11 depicts several paths planned by applying A^* with h_{max} in the top row and with h_{hyb} in bottom for identical

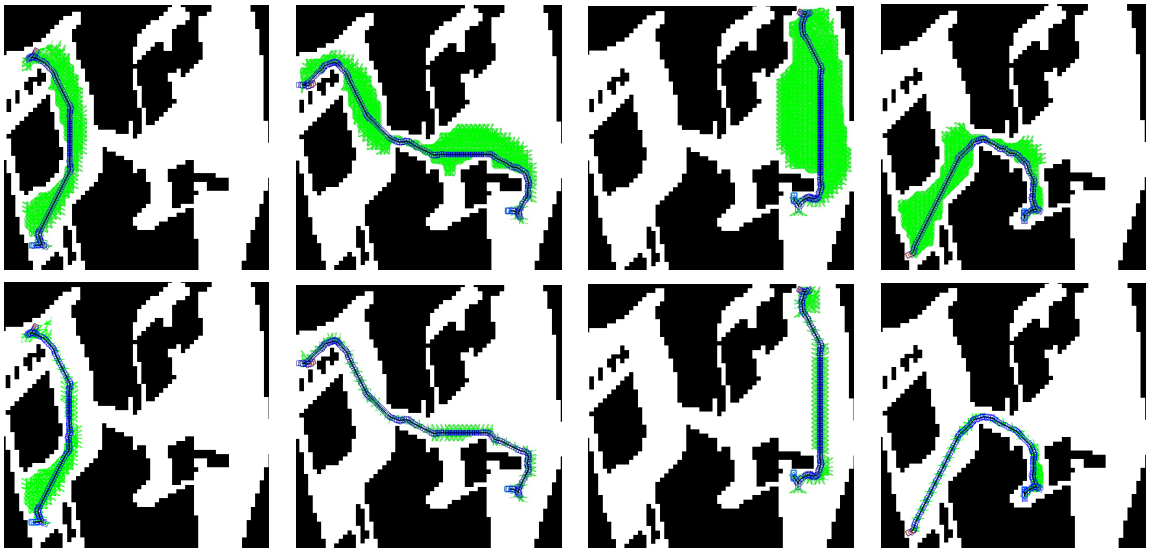
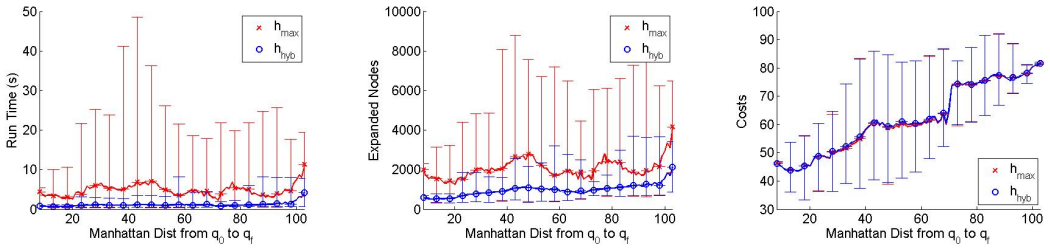


Fig. 11. Resulting paths by applying A^* with h_{max} (top) and h_{hyb} (bottom).



(a) (b) (c)

Fig. 12. Simulation performance depending on manhattan distance between the initial and goal states.

situations at each column. The map encodes Tampere University's Mobile Machine Lab area into a $50m \times 50m$ binary matrix with $0.5m$ resolution. The initial and goal states are randomly selected such that the initial state is invisible from the goal, since h_{hyb} is equivalent to h_{max} when the initial state is visible. Here, the threshold cost ρ was set to 5.

Fig. 12 shows the comparison of planning performance with h_{hyb} versus that with h_{max} for 15000 test cases. It highlights the benefits of applying the hybrid heuristic function. Runtime presented in Fig. 12(a) is slower than that of [2]. This is because the simulation in this paper has been implemented in Matlab which is higher-level than C++ used in [2]. However, h_{hyb} (blue circles) clearly outperforms h_{max} (red crosses) in terms of runtime (Fig. 12(a)) and expanded nodes (Fig. 12(b)). The data shows that planning with h_{hyb} is about five times faster than planning with h_{max}. The improvement in states expanded is greater than a factor of 1.5. On the other hand, planning with h_{hyb} provides slightly worse path costs than planning with h_{max}, but with less than a factor of 1.02, as shown in Fig. 12(c).

V. Conclusion

This paper proposes a new efficient heuristic estimate function to speed up non-holonomic motion planning. The heuristic is computed in three steps: 1) dividing cartesian space into visible and invisible spaces, 2) copying pre-computed non-holonomic-without-obstacle heuristics to the visible states, and 3) running dynamic programming on the invisible states by accumulating the visible heuristics with holonomi-with-obstacle costs. The numerical simulations demonstrate that the hybrid heuristic leads to an order of magnitude performance improvement, compared to existing heuristics [1] and [2].

References

[1] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research (IJRR)*, 28:933–945, 2009.

[2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Practical search techniques in path planning for autonomous driving. In *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, Chicago, USA, June 2008.

[3] R. Knepper and A. Kelly. High performance state lattice planning using heuristic look-up tables. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3375–3380, October 2006.

[4] S. Koenig and M. Likhachev. D* lite. In *the AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483, 2002.

[5] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. pages 1177–1183, 2007.

[6] M. Pivtoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pages 3231–3237, August 2005.

Short term path planning using a multiple hypothesis evaluation approach for an autonomous driving competition

Miguel Oliveira[1] Vitor Santos[1] and Angel D. Sappa[2]

Abstract— **This paper describes a practical implementation of short term path planning in autonomous navigation in unmapped or unstructured environments. Path planning is performed by generating multiple hypothesis of paths for the robot and then evaluating the quality of each path. In very dynamic environments, long term path planning is generally not very useful, so this paper embraces the approach of short term path planning and continuously revises the path plan and motion parameters after the perception from its onboard sensors. The solution has been applied to small scale robots that compete in an autonomous driving competition. These robots have won the last six editions of this competition.**

I. INTRODUCTION

One of the core problems in autonomous navigation is to calculate the correct steering of the robot. This is generally based on primary directives, such as to move to a GPS waypoint inside a corridor as in the case of the DARPA Grand Challenge *http://www.darpa.mil/grandchallenge05/*, or to follow the road to a certain objective, which was the case in the DARPA Urban Challenge *http://archive.darpa.mil/grandchallenge/*, just to mention two of the most remarkable full scale autonomous navigation challenges in recent years. Given these, or other, high-level directives, a path planner must first analyze the data from the sensors (or higher level representations of those created by perception modules) and then generate the appropriate path according to the restrictions streaming from the sensorial information.

In this paper, we present a path planning which was tested in an autonomous driving robotic competition that takes place every year in Portugal, within the National Robotics Open (http://robotica2011.ist.utl.pt/).

The Autonomous Driving Competition takes place in a road like scenario. The road has an 8-shape configuration delimited with two white lines which simulates a two-way road. For path planning algorithms, this competition is a challenge since that robots must not only navigate on the road but also, at the same time, avoid obstacles placed arbitrarily on unknown positions, handle tunnels where lack of light may disrupt vision based road detection, and cope with road maintenance areas that bring the detour from the road. Robots must cope with these obstacles and navigate the scenario as fast as possible. The complete rules and specifications of the competition may be found in http://robotica2011.ist.utl.pt/docs/2011_Conducao_Autonoma-regras-en.pdf. Figure 1 shows some images of this competition.

In general, path planning has been thoroughly studied by the robotics community. In [1] and [2], extensive reviews on this topic are provided. However, while the state of the art in path planning in general has reached a high level of maturity, its adjustment to the problem at hand is cumbersome. In fact several details make most of the classic path planning methods unfit to tackle the Autonomous Driving Competition, namely: (*i*) The fact that the robot travels at high speed (in relation to the scale of the scenario) demands that the path planning is fast to process; (*ii*) The reduced field of view of the robot (2 to 3 meters to the front) discards the usage of complex paths based on splines [2] [3], clothoids [4], or others; (*iii*) The fact that classic obstacle avoidance techniques such as Vector Field Histograms or dynamic window approach [5] do not account for the non-holonomic nature of a car-like robot; (*iv*) The inexistence of a global map (or of a large enough local one) in addition to the fact that the information present in it is sparse discards techniques based on occupation grids and similar techniques (most of the cells would have an unknown state).

Because of these specifications, we have decided to devise and develop an algorithm that chooses the best steering direction for the robot. In this sense, this is not a path planning algorithm but more a local, short term evaluator of the best steering angle for a robot. The algorithm starts by generating a set of possible paths using a non-holonomic vehicle model for the robot. Each path represents a possible heading of the robot. To account for simplicity and fast processing, these paths are first order curves, i.e., circumference arcs. Second, the algorithm builds up a snapshot of the current view of the world appropriated for subsequent processing. It does this by creating a desired path on the desired lane of the road in the form of a list of position and heading values, the attractor points. These points are not connected using splines or other methods. They can be viewed as beacons sparsely positioned on the lane, indicating the preferred position and heading of the robot. At the same time, laser obstacles are represented as a list of repelling points.

The rest of the paper is organized as follows: section II describes the perception algorithms; section III shows how several hypothetic paths can be generated using a few number of control parameters; section IV explains the generation of navigation markers; section V describes how paths are evaluated and the best path is selected; finally, sections VI

[1]M. Oliveira and V. Santos are with the Departement of Mechanical Engineering, University of Aveiro,Portugal mriem@ua.pt vitor@ua.pt

[2]A. Sappa is with the Computer Vision Center, Barcelona, Spain, asappa@cvc.uab.es

(*a*) (*b*)

Fig. 1. The Autonomous Driving Competition environment is a 8-shaped road like scenario. *a*) obstacle avoidance *b*) navigation on a road maintenance area.

and VII present results and conclusions.

II. Perception

The path planning is performed in a two dimensional space, coincident with the road plane. In order to obtain sensorial information mapped onto this reference frame, the Inverse Perspective Mapping (IPM) technique is employed. The IPM technique consists of transforming the images taken into a new reference frame where the perspective effect is corrected. This reference frame is usually defined on the road plane, so that the resulting images become a top view of the road. One of the advantages of IPM is that the subsequent perception algorithms can be computed in a 2D synthesized world, which significantly eases the tuning of convolution filters size [6], the stability of neural network's inputs [7], or the detection of features of interest [8]. The technique requires some a priori knowledge, namely, the geometric transformation relating the cameras' and road reference frames. This is equivalent to state that the camera's position, orientation and intrinsic parameters must be known before hand. Commonly, the IPM technique also assumes that the road ahead is flat, that is, all pixels from the input image are views of points in the real world from the XoY plane of the road's reference frame. This assumption is a core issue of IPM. If undertaken by mistake, due to the presence of other vehicles, pedestrians, obstacles, or steep slopes in the road, the IPM produces wrong representations in the undistorted image. This problem is addressed by several researchers on this field [9][10][11]. The authors own implementation of an extension to IPM using a laser range finder (LRF) is presented in [12]. Using this setup we also have the LRF data corrected to the IPM reference frame, as shown in Fig. 2.

The LRF unit's scan plane is parallel to the road's surface. Therefore, the laser scan points are also defined in the same coordinate system as the IPM image. Using the method proposed [12], the raw laser data is compressed into a list of obstacles, each containing a portion of the laser raw data points.

Lane marker detection is based on previous work, described in detail in [13]. The core idea is to derive a set of candidate lane markers through the analysis of the inverse perspective mapped image's brightness profile. These candidates are then characterized by a set of meaningful statistical descriptors. In the perspective effect free image, parameters like lane marker width and ratio of curvature change are well defined. These assumptions are employed to provide the actual lane marker detection. Figure 2 shows an example of lane marker detection on an IPM image.

In sum, the path planning algorithm will receive the position of the lane markers and of laser obstacles from the perception modules. Lane markers and laser obstacles are all registered in the instantaneous vehicle reference frame.

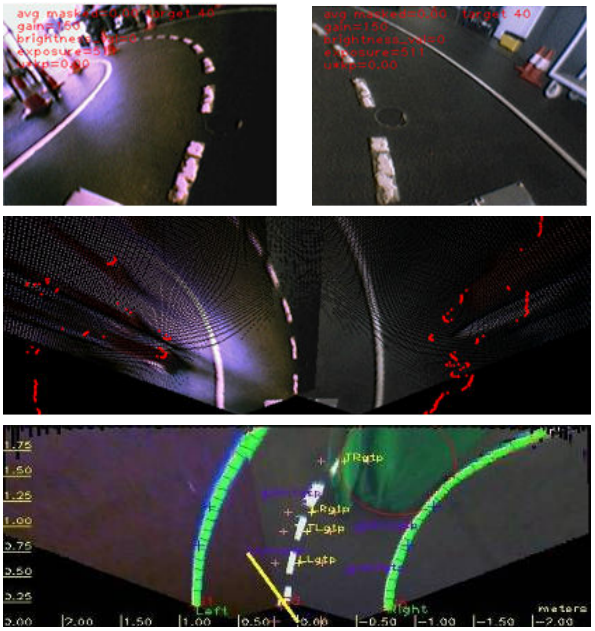


Fig. 2. (*top*) The images taken by the two camera onboard the robots; (*middle*) the image obtained using inverse perspective mapping, marked with referenced laser scan points (red dots); (*bottom*) lane marking detection example. Lane markers are highlighted in green.

III. Path generation

This paper uses two kind of descriptors for computing the optimum trajectory. These descriptors are in one case defined as a unitary vector and in the other as a 2D point. All descriptors are defined in the instantaneous vehicle reference system. Bold symbols will represent unitary vectors, i.e., $\mathbf{X} = [x \quad y \quad \theta]$, while 2D points, will be represented as $\mathtt{X} = [x \quad y]$. Throughout the paper we will use the sub-indices notation to refer to the components of the vector or point. For example, $\mathbf{X}_\theta$ refers to the angle θ of vector $\mathbf{X}$, while $\mathtt{X}_y$ refers to the y component of point $\mathtt{X}$. Left super indexes will notate the trajectory to which the variable belongs to. Right super indexes notate the index of the variable.

This section introduces the non-holonomic vehicle model used to generate a set of possible trajectories for the robot. It discusses how paths are generated and represented by nodes, using several high level criteria. From the observation of Fig. 3 it is clear that:

$$R = \frac{D}{tan(\alpha)}. \tag{1}$$

Hence, based on the steering direction α, it is possible to calculate the path that will be executed by the vehicle. Let ζ represent a path. The radius R of the instant center of rotation *ICR*. Given the arc length A, it is possible to calculate the angle β by:

$$\beta = \frac{A}{R}. \tag{2}$$

Let a unitary vector Π represent the Cartesian coordinates that lie on path ζ. The coordinates are given by:

$$\Pi = \begin{bmatrix} \Pi_x \\ \Pi_y \end{bmatrix} = \begin{bmatrix} R \cdot sin(\beta) \\ R - R \cdot cos(\beta) \end{bmatrix}, \tag{3}$$

combining equations (1), (2) and (3) results in:

$$\begin{bmatrix} \Pi_x \\ \Pi_y \end{bmatrix} = \begin{bmatrix} \frac{D}{tan(\alpha)} \cdot sin\left(\frac{A \cdot tan(\alpha)}{D}\right) \\ \frac{D}{tan(\alpha)} \cdot \left(1 - cos\left(\frac{A \cdot tan(\alpha)}{D}\right)\right) \end{bmatrix}. \tag{4}$$

For paths steering to the right, i.e., when $\alpha < 0$, the value of Π_y becomes symmetric of what is given in (4). The expression is only valid for $\alpha \in [-90, +90]$, which is perfectly adequate for most of the vehicles. Equation (4) is used to generate all paths and path nodes. The planner generates a set of paths, each is referred to as ${}^j\zeta$. For computation simplification purposes, each path is segmented into linear pieces and represented by a set of m nodes, notated as ${}^j\Pi \in \{{}^j\Pi^0, {}^j\Pi^1, ..., {}^j\Pi^m\}$. The discrete array of paths and their nodes are defined by setting values for the number of paths, the wheel angle of the first path α_0 and the angular spacing between paths ($\Delta\alpha$). Hence, the steering angle of a path ${}^j\zeta$ will be defined by:

$${}^j\alpha = {}^0\alpha + j \cdot \Delta\alpha. \tag{5}$$

The number of nodes for each path is defined by specifying the number of nodes and the arc length (A) between consecutive nodes. The *ith* node ${}^j\Pi^i$ will have an associated arc length segment given by:

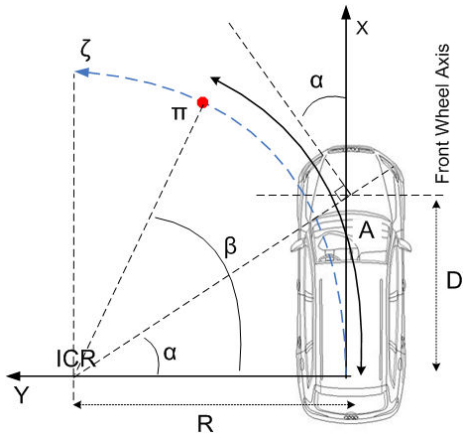


Fig. 3. The non-holonomic model of the vehicle is used to define the paths the robot will execute as a function of the angle imposed on the steering wheels.

$${}^jA^i = i \cdot A. \tag{6}$$

Applying ${}^j\alpha$ and ${}^jA^i$ into equation (4) provides the (x,y) coordinates of all path nodes. To determine the orientation of the node we compute the angle formed between the line segment defined by the current and the next node, and the vertical direction. Hence, the orientation of node i, ${}^j\Pi^i_\theta$ is given by:

$${}^j\Pi^i_\theta = atan\left(\frac{{}^j\Pi^{i+1}_y - {}^j\Pi^i_y}{{}^j\Pi^{i+1}_x - {}^j\Pi^i_x}\right). \tag{7}$$

Figure 4 shows an example of a set of paths and their nodes. Although the trajectories, the number of nodes in each trajectory, the $\Delta\alpha$ and A may change in runtime, they are considered equal for all paths, that is, all paths are equally spaced in wheel angle and will have the same number of nodes, which are in turn equally spaced by a constant arc length. Paths and their nodes are uniformly distributed across the provided limits. The methodology here proposed allows the user (or other higher level processes) to easily generate a large set of trajectories, using these small number of parameters. Different road scenarios have distinct requirements in terms of possible paths. For example a highway scenario, where a vehicle is traveling at high speed surely requires long trajectories to be tested, but the angular span of the vehicle's steering wheel is limited: in this case, a large value for A and number of nodes would generate long trajectories, while a small number of trajectories with the proper value for $\Delta\alpha$ would create a small angular span. In urban scenarios, small speed and hard turns are required: a small value of A will create short paths, while a large number of trajectories would provide trajectories with hard turns.

IV. Navigation Markers

As seen in section II, there are several perception modules running in parallel. When a perception module identifies a laser obstacle or the position of a lane marker, this information is is sent to the path planner module. However,

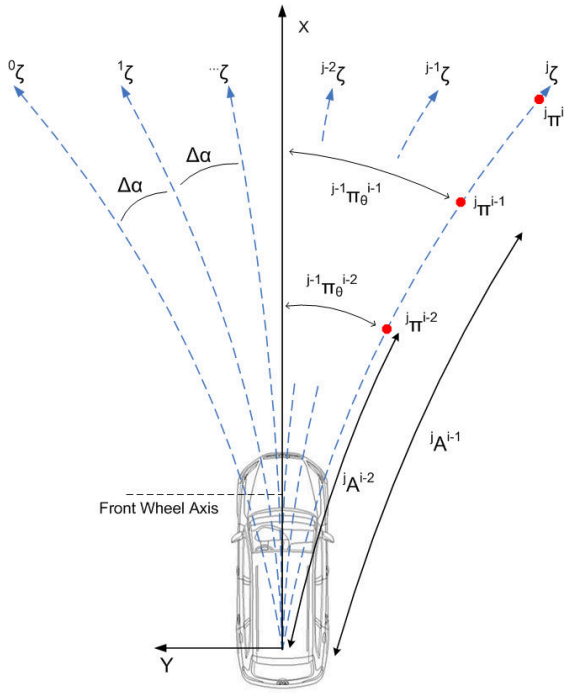


Fig. 4. An example of a set of paths defined using equally spaced wheel angle and arc length between nodes.

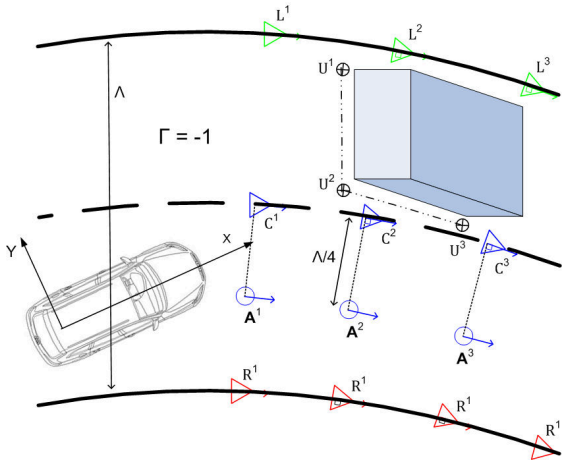


Fig. 5. The several navigation markers for a common road scenario. The road is described by the left, central and right points, $\mathtt{L}^k$, $\mathtt{C}^k$ and $\mathtt{R}^k$, represented by the triangles in the figure. The attractor points ($\mathbf{A}^k$, represented as circles in the figure) are generated using the central lane marker points combined with the road width (Λ) and the desired lane positioning behavior directive (Γ). Laser obstacles generate repelling points ($\mathtt{U}^k$) and are represented by circles with a cross.

the position of obstacles or lane markers must be represented in a way that is useful for the path planning module. This section describes this process of converting perception data into descriptions that are suited to be processed by our path planning algorithm, which are called Navigation Markers. Navigation Markers are computed from a given description of the scenario around the robot, i.e. from the information provided by the perception modules, and from a set of driving directives, which define what should be the navigation behavior. They are then used to compute several scores that will describe how good is a path for the current scenario. This process is an intermediate layer between the perception of the environment and the subsequent evaluation of all paths and selection of the optimum path.

There are two types of Navigation Markers: repelling and attractor points. Repelling points are computed after the laser obstacles positions generated using the data reduction algorithm (see section II). They have no information regarding orientation. For a given iteration of the path planner, there will be a list of $\mathtt{U}$ repelling points, defined as:

$$\mathtt{U}^k = [\mathtt{U}^k_x, \mathtt{U}^k_y]. \quad ,\forall k = 0,...,N, \tag{8}$$

where N is the number of repelling points.

Attractor points are, on the other hand, generated after the feature detectors that produce information regarding the road's positioning. The lane marker detection algorithms output a description of the road. This description consists of representing each of the three lane markings (left, central and right) as a list of points in the robot's reference frame. Let $\mathtt{L}^k = [\mathtt{L}^k_x, \mathtt{L}^k_y]$, $\mathtt{C}^k = [\mathtt{C}^k_x, \mathtt{C}^k_y]$ and $\mathtt{R}^k = [\mathtt{R}^k_x, \mathtt{R}^k_y]$ represent the lane marking descriptions. While the repelling point's position is obtained directly from the obstacles present in the laser scans, the location of the attractor points is dependent on the driving directive. The path planner module is informed constantly by another higher level process of the desired lane positioning behavior (Γ). This directive is defined as: $\Gamma = -1$, to drive on the left lane, $\Gamma = 0$, to drive on the center of the road and $\Gamma = 1$, to drive on the right lane. Since all the lane marker detection modules produce a standardized description of the road, the information of the left central and right lane markers is for now redundant. Currently, we use the central lane marker's description $\mathtt{C}$ to generate the attractor points. Let the unitary vector $\mathbf{A}$ notate the attractor points. Using the directive Γ combined width the road width Λ (assumed to be known a priori) the attractor points are defined as:

$$\mathbf{A}^k = \begin{bmatrix} \mathbf{A}^k_x \\ \mathbf{A}^k_y \\ \mathbf{A}^k_\theta \end{bmatrix} = \begin{bmatrix} \mathtt{C}^k_x + \dfrac{\Gamma \cdot \Lambda}{4} \cdot cos(\mathbf{A}^k_\theta) \\ \mathtt{C}^k_y + \dfrac{\Gamma \cdot \Lambda}{4} \cdot sin(\mathbf{A}^k_\theta) \\ atan\left(\dfrac{\mathtt{C}^k_y - \mathtt{C}^{k-1}_y}{\mathtt{C}^k_x - \mathtt{C}^{k-1}_x}\right) \end{bmatrix}. \tag{9}$$

Figure 5 depicts the Navigation Markers in a typical right turn scenario.

V. Path Evaluation

Once Navigation Markers are generated after the perception and several possible paths have been computed, the goal now is to find which of those paths is the more adequate for the current scenario. To perform this task, several evaluation functions Ω are employed. Each evaluation function returns normalized score $\hat{\Omega}$ that ascertains how well the path suits the criteria evaluated by the function. There are four evaluation functions, which will be described bellow. In many of the following evaluations it will be necessary to compute, for a given trajectory node Π^i, the closest navigation marker:

for example, what is the closest attractor point to a certain trajectory node. Let function $map(i)$ be the function that retrieves the index k of the closest navigation marker to node i. It is defined as:

$$map(i) = argmin_k \left(\sqrt{ \left({}^j\Pi_x^i - \mathbf{A}_x^k \right)^2 + \left({}^j\Pi_y^i - \mathbf{A}_y^k \right)^2 } \right). \quad (10)$$

A. Ω^1: Average distance to attractor points

This criteria measures a path's average distance to the attractor points and evaluates how close a path is to the attractor points. A perfect score means that the path is coincident with all the attractor points. For each path ${}^j\zeta$, the criterion is calculated as a function of the average distance between all path nodes and the closest attractor point to each node:

$$\Omega^1({}^j\zeta) = \frac{\sum_{i=0}^{N} \left(\sqrt{ \left({}^j\Pi_x^i - \mathbf{A}_x^{map(i)} \right)^2 + \left({}^j\Pi_y^i - \mathbf{A}_y^{map(i)} \right)^2 } \right)}{N}, \quad (11)$$

where N is the number of trajectory nodes. The normalized evaluation score $\hat{\Omega}^1$ is obtained using the maximum admissible value as a normalizing factor. In our implementation, this value is set as the road's width Λ.

$$\hat{\Omega}^1({}^j\zeta) = 1 - \frac{max\left(\Omega^1({}^j\zeta), \Lambda \right)}{\Lambda}. \quad (12)$$

Figure 6 shows an example of the Ω^1 calculation.

B. Ω^2: Average angular difference to attractor points

This criterion measures the path average angular difference to the attractor points (Ω^2) and evaluates how compliant the path is to the attractor points heading. A perfect score means that the path drives the robot along the direction defined by the attractor points. For each path ${}^j\zeta$, the criterion is calculated as follows by measuring the average angular difference between each node's orientation and the closest attractor point's orientation:

$$\Omega^2({}^j\zeta) = \frac{\sum_{i=0}^{N} |({}^j\Pi_\theta^i - \mathbf{A}_\theta^{u(i)})|}{N}, \quad (13)$$

where N is the number of nodes of the path. To obtain the final score for this evaluation a normalization is computed using the maximum admissible angular difference. In our case, we set this value to 180 degrees.

$$\hat{\Omega}^2({}^j\zeta) = 1 - \frac{max\left(\Omega^2({}^j\zeta), \pi \right)}{\pi}. \quad (14)$$

Figure 7 depicts how $\Omega^2({}^j\zeta)$ values are computed.

C. Ω^3: Average laser obstacle clearance

This criterion measures a path's average distance to the laser obstacles repelling points (Ω^3). It evaluates how close path ${}^j\zeta$ is to the laser obstacles repelling points $\mathtt{U}^k$.

$$\Omega^3({}^j\zeta) = \frac{\sum_{i=0}^{N} \mathbf{f}(i)}{N}, \quad (15)$$

where N corresponds to the number of nodes and $\mathbf{f}(i)$ is a function that avoids local minima that typically occurs in the middle of two obstacles, the minimum distance from each path node to all repelling points is compared to a predefined variable that defines the saturation threshold for the obstacle clearance Φ.

$$\mathbf{f}(i) = \begin{cases} \mathbf{d}(i), & if \quad \mathbf{d}(i) > \Phi \\ 0, & otherwise \end{cases}, \quad (16)$$

where $d(i)$ is a function that retrieves the value of the distance from the path node i to the closest repelling point:

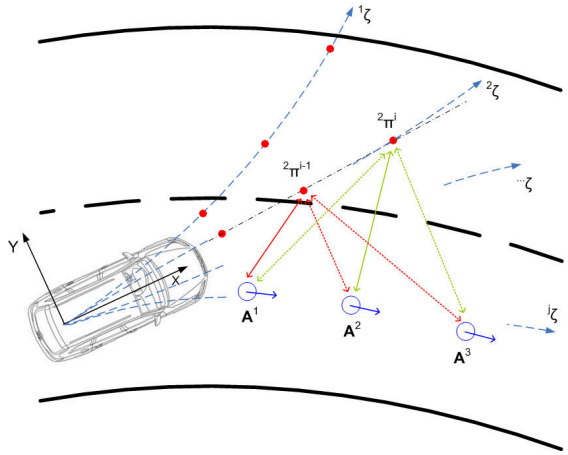


Fig. 6. An example of the calculation of Ω^1 for nodes ${}^2\Pi^{i-1}$ and ${}^2\Pi^i$. For each node, the minimum distance to all attractor points is selected (represented with a solid line). Then the average of these minima is calculated. In this figure the Ω^1 criteria will have the following score arrangement: $\hat{\Omega}^1({}^j\zeta) > \hat{\Omega}^1(\cdots\zeta) > \hat{\Omega}^1({}^2\zeta) > \hat{\Omega}^1({}^1\zeta)$.

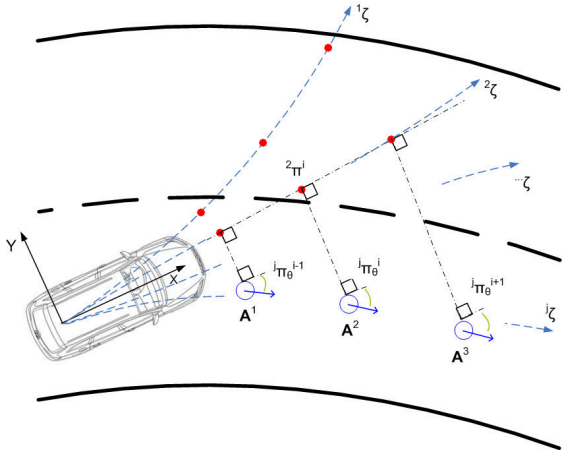


Fig. 7. Calculation of Ω^2 scores. For each node, the minimum angular difference to the closest attractor points is selected. Then the average of these minima is calculated. In this figure, because higher index trajectories suit better the attractor points heading, the following occurs: $\hat{\Omega}^2({}^j\zeta) > \hat{\Omega}^2(\cdots\zeta) > \hat{\Omega}^2({}^2\zeta) > \hat{\Omega}^2({}^1\zeta)$.

$$\mathbf{d}(i) = \sqrt{\left({}^{j}\Pi_x^i - \mathrm{U}_x^{map(i)}\right)^2 + \left({}^{j}\Pi_y^i - \mathrm{U}_y^{map(i)}\right)^2}. \quad (17)$$

The normalized score is obtained as usual:

$$\hat{\Omega}^3({}^{j}\zeta) = 1 - \frac{max\left(\Omega^3({}^{j}\zeta), \Phi\right)}{\Phi}. \quad (18)$$

Figure 8 shows an example of the calculation of this evaluation criteria.

D. Ω^4: Free space

The final evaluation criteria is used to guarantee that the selected path is collision free. This is done by measuring the possible collisions of a path with the laser obstacles. In order to do so, a poly line representation of the robot for each path node is created using the width of the robot as a reference. As described, laser obstacles are defined by a set of points. These points were obtained using the data reduction scheme introduced in section II. Figure 9 shows a laser obstacle composed of three points, U^1, U^2 and U^3. A set of line segments for each path node ${}^{j}\Pi^i$ is defined as ${}^{j}\Upsilon^i$. In Figure 9, set ${}^{2}\Upsilon^1$ is composed of the lines: $\overline{P_1P_2}$, $\overline{P_2P_3}$, $\overline{P_3P_4}$, $\overline{P_4P_1}$ and also of the line $\overline{{}^{2}\Pi^{1}{}^{2}\Pi^{2}}$. The free space analysis consists of searching for intersections between the lines defined by consecutive laser obstacle points with the lines in the set ${}^{j}\Upsilon^i$:

$$\Omega^4({}^{j}\zeta) = {}^{j}A^u, \quad (19)$$

where ${}^{j}A^u$ is the arc length of node index u (see eq. (6)), and u is the index of the node with maximum arc length that does not collide with an obstacle:

$$u = \begin{cases} argmax_i\left(\{{}^{j}\Pi^i\}\right), & if \neg intr(\overline{L_{i-1}L_i}, \overline{\Upsilon^i}) \\ 0, & otherwise \end{cases}, \quad (20)$$

where $intr$ is a function that tests for the intersection of the two groups of line segments and returns true if an intersection occurs. The evaluations returns maximum arc length distance that each path may accomplish before it collides with an obstacle. If the trajectory is collision free, the arc length of the most distant node is returned. Since we employ only first order paths, sometimes a path that will lead to a collision very far away from the robot may still be interesting to follow, up to a certain moment. In these cases, the path should not be discarded immediately. To handle this, the free space evaluation employs the notion of minimum safety distance Ψ which can be ascertained as the minimum arc length distance for a path where no collisions occur, in order to validate this path. The normalized value of this criteria is given as:

$$\hat{\Omega}^4({}^{j}\zeta) = \begin{cases} 1, & if \quad \Omega^4({}^{j}\zeta) > \Psi \\ 0, & otherwise \end{cases} \quad (21)$$

E. Overall score

The final score of a path is defined using a weighted average of the three first evaluations. In the case of the free space analysis, it is used to discard a path that will lead to a collision. We propose the following expression for obtaining a path overall score (${}^{j}\hat{\zeta}$).

$${}^{j}\hat{\zeta} = \hat{\Omega}^4({}^{j}\zeta) \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \cdot \begin{bmatrix} \hat{\Omega}^1({}^{j}\zeta) \\ \hat{\Omega}^2({}^{j}\zeta) \\ \hat{\Omega}^3({}^{j}\zeta) \end{bmatrix}, \quad (22)$$

where w_i is the weight corresponding to each criteria. In order to have a normalized overall score, the weights must add up to 1. The ratio between w_1 and w_2 defines the reactivity of the path, i.e., when $w_1 >> w_2$ the paths that rapidly bring the robot close to the attractor points will have better scores. The trade off is that the robot may be close to the desired position but without the desired orientation. On the other hand, if $w_1 << w_2$ then the best scored paths

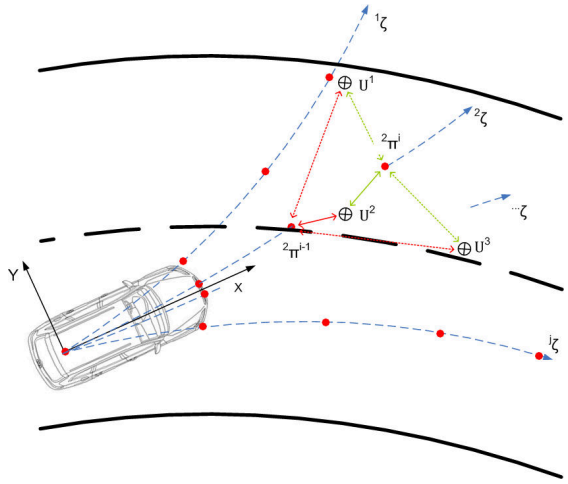


Fig. 8. Calculation of Ω^3 scores. For each node, the minimum distance to the closest repelling points is selected. Then the average of these minima is calculated. In this figure the $\hat{\Omega}^3({}^{j}\zeta)$ score will be the highest of all trajectories, since path ${}^{j}\zeta$ is the one with best laser obstacle clearance.

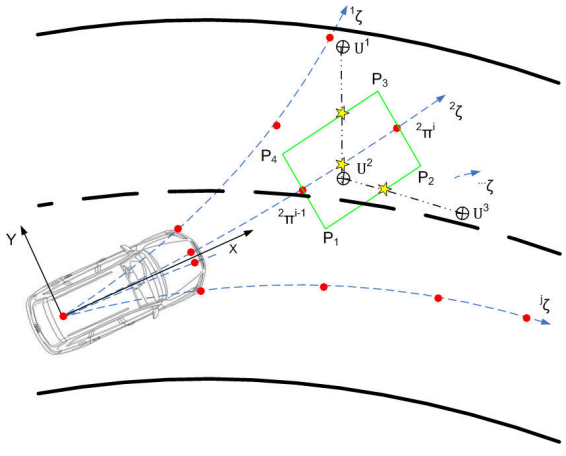


Fig. 9. Poly line representation of the robot using the robot width as an estimate for defining several line segments. They are then tested for intersection with the lines defined by consecutive laser obstacles repulsor points U. In this case, path ${}^{2}\zeta$ intersects with the laser obstacle (marked by stars), which leads to a score $\hat{\Omega}^4({}^{j}\zeta) = 0$.

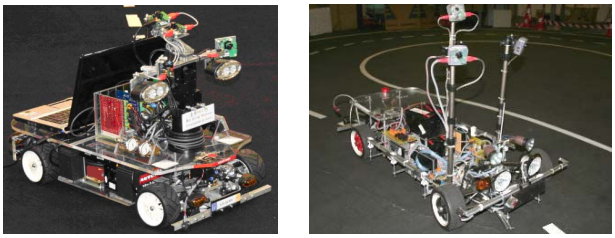


Fig. 10. The ATLAS robots have won the Autonomous Driving Competition of the National Robotics Open from 2006 to 2011.

will have an orientation similar to that of the attractor points, and the algorithm will bring the robot smoothly to the attractor points but with more guarantees of having the desired orientation. The selection of the path to impose on the robot is obtained by finding the highest score amongst all paths. These weights where empirically tuned as presented in next section.

VI. Results

The path planning algorithm described in the previous sections was successfully applied to the small scale robots of the Atlas project http://atlas.web.ua.pt/ depicted in Fig. 10. These robots are equipped with several cameras and a LRF. The parameters of the motion planning algorithm were tuned empirically. The values are of course setup specific. They are dependent on the scale of the robots as well as other characteristics such as velocity or maximum wheel turn angle. However, we found that tuning these parameters is not a very difficult task. They have meaningful functionalities and clear influence on the robot's navigation behavior. This helps an user to quickly find a good compromise for the values of these parameters. In fact, the same path planning algorithm was used for the two distinct robots in Fig. 10 without requiring much effort in tunning the parameters. For reference, Table I shows the values of the parameters that are used in one of the robots.

Figure 11 shows some examples of the path planner in obstacle free environments. Figure 12 shows the performance of the path planner in a scenario populated by obstacles. In both cases the algorithm is able to select the most adequate path.

The algorithm has been thoroughly tested in the ATLAS robots during the Autonomous Driving Competition of the National Robotics Open. The Atlas robots have won the last six editions of the competition, i.e., from 2006 to 2011. This shows that these robots are very capable of navigating in complex scenarios. The presented path planing algorithm is a cornerstone of these capabilities.

The path planning algorithm is computed in less than 5 miliseconds (on a Dual Core 2.5GHz HP 8510p) which enables real time execution.

Several videos of the robots in different competitions and laboratory tests can be found at http://lars.mec.ua.pt/public/Media/path_planning/. From these it is possible to evaluate the navigation capabilities of the Atlas robots. Several situations are shown:

TABLE I

Path planning parameters values used on the AtlasMV.

Parameter	Equation	Value on AtlasMv
Φ	(18)	110% of half the robot's width
Ψ	(21)	1.5 meters
w_1	(21)	0.25
w_2	(21)	0.25
w_3	(21)	0.5

- Navigation on the road, tunnel and road maintenance area, during the 2010 competition;
- Fast speed navigation with obstacle avoidance, during the 2011 competition;
- Very fast speed navigation with no obstacles, during the 2009 competition.
- One robot pursuing another. The front robot is being remote controlled while the second follows it using laser data. Demo at the 2009 competition;
- Obstacle avoidance with reverse maneuver. During the 2006 competition;
- Tunnel navigation, tests at the 2011 competition;
- A view of several paths being tested in real time;
- Inverse perspective mapping using two cameras mounted on a pan and tilt system;
- Lane marker detection and navigation: robot's view;
- Driving directive: driving on the left lane;
- Driving directive: driving on the right lane;
- Fast speed driving on the right lane.

VII. Conclusions

This paper presents an integrated solution for the steering calculation of an autonomous robot. The architecture is entirely modular: first, sensors including laser and cameras are registered into a common reference frame using multi-camera inverse perspective mapping [14]; then several detector modules process the fused sensorial information looking for patterns of interest for the navigation or for laser obstacles; finally, representations of the detected patterns are passed to the algorithm which decides the most adequate path. The path planning module starts by generating a discrete set of possible paths using a non-holonomic vehicle model. This model reduces the search space of the possible paths, therefore improving computational performance. Then, using the proposed criteria, paths are evaluated against attractor and repelling points. These criteria are then fused and the highest scoring path is used to decide the adequate steering. The presented algorithm has been incorporated in the Atlas robots and has played an important role in the winning of the last six editions of the autonomous driving competition. The path planner algorithm has worked flawlessly during many hours of autonomous driving. The next step will be to test the algorithm in real world scenarios to assess performance.

Acknowledgement

This work was supported by the Portuguese Foundation for Science and Technology under grant SFRH/43203/2008

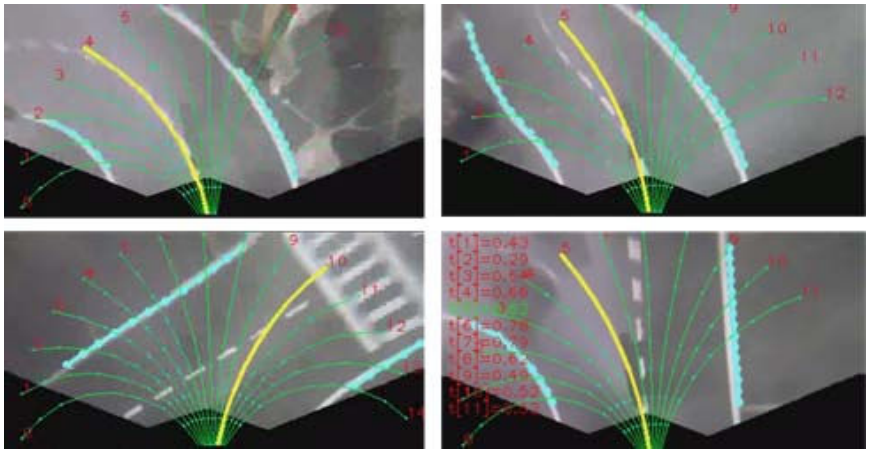


Fig. 11. Some examples of path evaluation. In this case, the road is detected using the technique described in III.B. The paths are represented by the green arcs. The highest scoring path is highlighted in yellow.

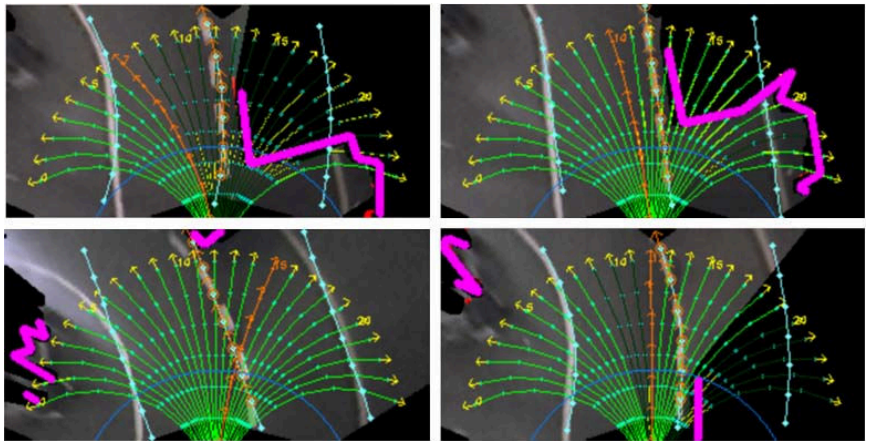


Fig. 12. Some examples of path evaluation in obstacle populated scenarios. The paths are represented by the green arcs. The highest scoring path is highlighted in orange, and the laser obstacles are represented by the thick magenta lines.

and the Spanish Government under project TIN2011-25606 and research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018).

REFERENCES

[1] J. P. Laumond, "Advanced holonomic and non-holonomic planning," *Robot Motion Planning and Control, Springer*, 1998.

[2] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments: Part i," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 1063 –1069.

[3] D. Ferguson, M. Howard, and M. Likhachev, "Motion planning in urban environments: Part ii," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 1070 –1076.

[4] D. Khosla, "Accurate estimation of forward path geometry using two-clothoid road model," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 1, june 2002, pp. 154 – 159 vol.1.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23 –33, mar 1997.

[6] J. McCall and M. Trivedi, "Performance evaluation of a vision based lane tracker designed for driver assistance systems," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, june 2005, pp. 153 – 158.

[7] D. Pomerleau, "Ralph: rapidly adapting lateral position handler," in *Intelligent Vehicles '95 Symposium., Proceedings of the*, sep 1995, pp. 506 –511.

[8] M. Bertozzi, A. Broggi, and A. Fascioli, "Real-time obstacle detection on a massively parallel linear architecture," in *Algorithms and Architectures for Parallel Processing, 1997. ICAPP 97., 1997 3rd International Conference on*, dec 1997, pp. 535 –542.

[9] M. Bertozzi and A. Broggi, "Gold: a parallel real-time stereo vision system for generic obstacle and lane detection," *Image Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 62 –81, jan 1998.

[10] A. Broggi, S. Cattani, P. P. Porta, and P. Zani, "A laserscanner-vision fusion system implemented on the terramax autonomous vehicle," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 111 –116.

[11] R. Labayrade, C. Royere, D. Gruyer, , and D. Aubert, "Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner," *Image Processing, IEEE Transactions on*, vol. 19, pp. 117 –140, 2005.

[12] J. Almeida and V. Santos, "Laser-based tracking of mutually occluding dynamic objects," in *Portuguese Conference in Automatic Control. 2010*, Sep. 2010.

[13] M. Oliveira and V. Santos, "Real time extraction of road border lines using simple statistical descriptors," in *In 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2008 Intelligent Robots Systems, IEEE International Conference on*, sept. 2008.

[14] M. A. Oliveira and V. Santos, "Multi-camera active perception system with variable image perspective for mobile robot navigation,," in *Conference on Mobile Robots and Competitions, Portuguese Robotics Open.*, Apr. 2008.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session V

Perception & Situation awareness

- **Keynote speaker: Wolfram Burgard (Freiburg University, Frieburg, Germany)**
  Title: Localization and Mapping in Dynamic and Changing Environments
  Co-authors: Gian Diego Tipaldi

- **Title: Real-time Scan-Matching Using L0-norm Minimization Under Dynamic Crowded Environment**
  Authors: Y. Hieida, T. Suenaga, K. Takemura, J. Takamatsu and T. Ogasawara

- **Title: Fast classification of static and dynamic environment for Bayesian Occupancy Filter (BOF)**
  Authors: Q. Baig, M. Perrollaz, J. Botelho Do Nascimento, C. Laugier

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session V

Keynote speaker: **Wolfram Burgard (Freiburg University, Frieburg, Germany)**

Localization and Mapping in Dynamic and Changing Environments

Co-authors: Gian Diego Tipaldi

Abstract: The majority of existing approaches to mobile robot localization and mapping assumes that the world is static, ignoring the dynamics inherent in most real world scenarios like parking lots, warehouses and even offices and households. In such environments the configuration of certain objects such as cars, goods, or furniture can change with time leading to inconsistent observations with respect to previously learned maps and thus decreasing the localization accuracy. In this talk we present a probabilistic grid-based approach for modeling changing environments. Our method represents both, the occupancy and its changes in the corresponding area where the dynamics are characterized by the state transition probabilities of a Hidden Markov Model. We further present a novel probabilistic approach to lifelong localization in changing environments, where the robot pose and the environment state are jointly estimated using a Rao-Blackwellized particle filter. Exploiting several characteristics of HMMs, we can considerably speed up the estimation procedure. This makes it feasible to run our algorithm on-line. Experimental results obtained with data acquired by real robots demonstrate that our model is well-suited for representing changing environments. Further results demonstrate that our approach can reliably adapt to changes in the environment and that it significantly improves standard localization techniques.

Biography: Wolfram Burgard is a professor for computer science at the University of Freiburg, Germany where he heads the Laboratory for Autonomous Intelligent Systems. He studied Computer Science at the University of Dortmund and received his Ph.D. degree in computer science from the University of Bonn in 1991. His areas of interest lie in artificial intelligence and mobile robots. In the past, Wolfram Burgard and his group developed several innovative probabilistic techniques for robot navigation and control. They cover different aspects including localization, map-building, path planning, and exploration. He received the prestigious Gottfried Wilhelm Leibniz-Preis in 2009, an advanced ERC grant in 2010 and he is a fellow AAAI since 2009.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Lifelong Localization and Dynamic Map Estimation in Changing Environments

Gian Diego Tipaldi[1] Daniel Meyer-Delius[2] Maximilian Beinhofer[1] Wolfram Burgard[1]

***Abstract*—Robot localization systems typically assume that the environment is static, ignoring the dynamics inherent in most real world scenarios like parking lots, warehouses and even offices and households. In such environments the configuration of certain objects such as cars, goods, or furniture can change with time leading to inconsistent observations with respect to previously learned maps and thus decreasing the localization accuracy. In this paper we present a novel probabilistic approach to lifelong localization in changing environments, where the robot pose and the environment state are jointly estimated using a Rao-Blackwellized particle filter. To describe the environment, we utilize a hidden Markov model formulation. Exploiting several characteristics of this model, we can considerably speed up the estimation procedure. This makes it feasible to run our algorithm online. Experimental results obtained with a real robot in a dynamically changing environment demonstrate that our approach can reliably adapt to changes in the environment and that it significantly outperforms standard localization techniques.**

I. Introduction

Long term operation of mobile robots in changing environments has become a major focus of interest in robotics research in recent years, as this ability is required for robots navigating in the real world. One of the most challenging tasks in this context is that of dealing with the dynamic aspects of the environment. Many existing approaches to robot navigation assume that the world is static and apply models which treat dynamic objects as outliers [1, 2, 3]. For highly dynamic objects like moving people or cars, these methods typically work quite well, but they are less effective for *semi-static* objects. By semi-static objects we mean objects which change their state slowly or seldom, like parked cars, doors, pallets in warehouses or furniture which can be moved. In many realistic scenarios (see Figure 1), in which robots operate for extended periods of time, semi-static objects are ubiquitous and ignoring them can substantially deteriorate the navigation performance.

In this paper, we present a novel approach to lifelong localization in changing environments, which explicitly takes into account the dynamics of the environment. The approach is able to distinguish among objects that presents fast dynamic behaviors, e.g., cars and people, objects that can be moved around and change configuration, e.g., boxes, shelfs, doors, and objects that are static and do not move around,

Fig. 1. A robot navigating in a parking lot at noon (left) and at 6 pm (right). Note that despite being at the same spot in both cases, the observations will be substantially different due to the changed number of parked cars.

e.g., walls. To represent the environment, we use a *dynamic occupancy grid* [4], which employs hidden Markov models on a two-dimensional grid to represent the occupancy and the corresponding transition probabilities for each cell of this grid. We first learn the parameters of this model using a variant of the expectation maximization (EM) algorithm and then use this information to jointly estimate the pose of the robot and the state of the environment during global localization. We employ a Rao-Blackwellized particle filter (RBPF), in which the robot pose is represented by the sampled part of the filter, and the occupancy probability of a cell is represented in the analytical part of the factorization. We further propose a map management method, which uses a local map representation that is able to forget changes in a sound probabilistic way, by considering the mixing times of the associated Markov chain, and to minimize memory requirements.

Compared to previous approaches, our algorithm has several desirable advantages. First, it improves the robustness and accuracy of the pose estimate of the robot. Second, our method is able to provide an up-to-date map of the environment. Finally, our map management method considerably reduces the runtime of the process whilst minimizing the memory requirements. As a result, our approach allows a robot to simultaneously perform the estimation of its pose and the potentially changing state of the environment in an online fashion. To the best of our knowledge, this is the first approach to address this problem in a general and systematic way. Previous attempts either focused on how to filter spurious observations due to dynamic objects, focused only on limited areas or individual elements of the map or specifically addressed the problem of pose tracking.

II. Related Work

Most mobile robot navigation systems rely on a map of the environment built beforehand in an offline phase. A popular approach to deal with subsequent changes in the environment

[1] G. D. Tipaldi, M. Beinhofer, and W. Burgard are with the University of Freiburg, Dept. of Computer Science, D-79110 Freiburg, Germany.

[2] D. Meyer-Delius is with the KUKA Laboratories GmbH, D-86165 Augsburg, Germany.

This work has been partially supported by the European Commission under contract numbers FP7-248258-FirstMM and FP7-260026-TAPAS. Also by the German Research Foundation (DFG) with in the Research Training Group 1103.

is to filter out sensor measurements caused by dynamic objects. Several approaches rely on probabilistic sensor models that identify the measurements that are inconsistent with the map. For example, Fox *et al.* [1] use an entropy gain filter and a distance filter based on the expected distance of a measurement, while Schultz *et al.* [2] uses minima of the laser scan together with an already available map. Orthogonal to the work on localization in dynamic environments, many authors have addressed the problem of modeling such environments.

Other approaches specifically focus on building two maps, one for the static part of the environment and one that accounts for dynamic objects. Wolf and Sukhatme [5] propose a model that maintains two separate occupancy grids, one for the static parts of the environment and the other for the dynamic parts. Wang *et al.* [6] formulate a general framework for mapping and dynamic object detection and developed a system to detect if a measurement is caused by a dynamic object. Montesano *et al.* [7] extends the approach by jointly considering the problem of dynamic object detection with the one of mapping, by including the error estimation of the robot in the classifier.

Biber and Duckett [8] propose a model that represents the environment on multiple timescales simultaneously. For each timescale a separate sample-based representation is maintained and updated using the observations of the robot according to an associated timescale parameter. Our extends it by providing a sound model of the change and the possibility of inferring the optimal timescale parameter for each grid cell. Those approaches, however, assumes that the position of the robot is known with a certain accuracy to compute and update the maps and therefore are not suited for global localization problems.

Murphy *et al.* [9] showed how a Rao-Blackwellized particle filter solution to the SLAM problem can deal with dynamic maps in a theoretical way. Their approach, however, is computationally too complex to handle real world scenarios and computes a map in the robot frame. This limits its applicability in real world scenarios and in a global localization settings, where the transformation between the world reference frame and the robot frame is not available. Avots *et al.* [10], for example, use a Rao-Blackwellized particle filter to estimate the pose of the robot and the state of doors in the environment. They represent the environment using a reference occupancy grid where the location of the doors is known, but not their state (i.e., opened or closed). Petrovskaya and Ng [11] propose a similar approach where instead of a binary model, a parametrized model (i.e., opening angle) of the doors is used. Similar to these approaches, we also use a Rao-Blackwellized particle filter to estimate the pose of the robot and the state of the environment. Stachniss and Burgard [12] estimate typical configurations of dynamic areas in the environment of a mobile robot. Their approach clusters local grid maps to identify the possible configurations and uses these clusters to localize a mobile robot in a non-static environment. Similarly, Meyer-Delius *et al.* [13] keep track of the observations caused by unexpected objects in the environment using temporary local maps. The robot pose is then estimated using a particle filter that relies both on these temporary local maps and on a reference map of the environment. Lately, an interesting approach to lifelong mapping in dynamic environments has been presented [14]. The approach focuses mainly on visual maps, and presents a framework where local maps (views) can be updated over time and new local maps are added/deleted when the configuration of the environment changed. In contrast to their methods, however, we estimate the state of the complete environment, and not only of a small, specific area or element. Additionally, we also learn the model parameters from data and we are able to generalize over unforeseen environment configurations.

III. Dynamic Occupancy Grid

Occupancy grids [15] are one of the most popular representations of a mobile robot's environment. They partition the space into rectangular cells and store, for each cell, a probability value indicating whether the underlying area of the environment is occupied by an obstacle or not.

One of the main disadvantages of occupancy grids is that they assume the environment to be static. To be able to deal with changes in the environment we utilize a *dynamic occupancy grid* [4], a generalization of an occupancy grid that overcomes the static-world assumption by explicitly accounting for changes in the environment. A dynamic occupancy grid relies on an HMM (see Rabiner [16]) to explicitly represent both the belief about the occupancy state and state transition probabilities of each grid cell.

We assume that the map consists of a collection of individual cells, $m_t = \{c_t^{(i)}\}$, following the standard occupancy grid assumptions, and each cell is modeled using an Hidden Markov Model (HMM). The state transition probabilities $p(c_t \mid c_{t-1})$ of each HMM describe how the occupancy state of cell c changes between consecutive time steps. Since a cell c can be either free (*f*) or occupied (*o*), the state transition model is specified using only two transition probabilities, namely $p(c_t = f \mid c_{t-1} = f)$ and $p(c_t = o \mid c_{t-1} = o)$. Note that assuming a stationary process for the changes in the environment, these probabilities do not depend on the absolute value of t. Standard occupancy grids are a special case of dynamic occupancy grids where the transition probabilities $p(c_t = f \mid c_{t-1} = f)$ and $p(c_t = o \mid c_{t-1} = o)$ are 1 for all cells c.

The estimation of the occupancy state of a cell follows a Bayesian approach according to

$$p(c_t \mid z_{1:t}) = \eta\, p(z_t \mid c_t) \sum_{c_{t-1} \in \{f,o\}} p(c_t \mid c_{t-1})\, p(c_{t-1} \mid z_{1:t-1}), \quad (1)$$

where $p(z_t \mid c_t)$ and $p(c_t \mid c_{t-1})$ correspond respectively to the observation and transition models of the cell and η is a normalization constant. The observation model specifies the likelihood of measuring a cell as occupied or free and is assumed to depend only on the sensor used and not on the location, therefore it is specified beforehand and is the same for each HMM. As explained in [4], the transition model and

the observation model for each cell is estimated from observed data using an instance of the expectation-maximization (EM) algorithm.

IV. Simultaneous Localization and Dynamic State Estimation

In this section we describe our approach to simultaneously estimate the robot pose and the dynamic state of the environment. Although on first sight one can see the addressed problem as an instance of the better known simultaneous localization and mapping (SLAM), there are two main differences between them.

The first difference is the absence of a global reference frame in the SLAM problem. No global pose is required and the initial pose of the robot can typically be set freely. On the contrary, we explicitly address global localization as part of the estimation aspect. We have a global reference frame and the initial pose of the robot is unknown and assumed to be uniformly distributed over the whole environment. The second difference regards the dimensionality of the map. In the SLAM problem, the size of the map is not known in advance and can grow unbounded with time. In our problem the size of the map is known and we only focus on estimating the actual configuration of the dynamic objects present in the environments. Despite the differences, the two problems do share the same state space, i.e., the robot pose and the state of the map, and one can exploit the same factorization that made Rao-Blackwellized particle filters a feasible solution to the SLAM problem [9, 17].

In the following we show how this factorization can be exploited and we derive the RBPF that will be used to estimate the posterior $p(x_t, m_t \mid z_{1:t}, u_{0:t-1}, m_{t-1})$ about the robot pose x_t and the configuration of the environment m_t, given the observations $z_{1:t}$, the odometry measurements $u_{0:t-1}$ and the prior over the map m_0, which is computed using the limiting distribution of the HMM. The key idea is to separate the estimation of the robot pose from the map estimation process,

$$p(x_t,m_t \mid z_{1:t}, u_{1:t-1}, m_{t-1}) = p(m_t \mid x_t, z_{1:t}, m_{t-1}) p(x_t \mid z_{1:t}, u_{1:t-1}, m_{t-1}). \quad (2)$$

This can be done efficiently, since the posterior over maps $p(m_t \mid x_t, z_{1:t}, m_{t-1})$ can be computed analytically given the knowledge of x_t, $z_{1:t}$ and m_{t-1} and using the forward algorithm for the HMM. The remaining posterior, $p(x_t \mid z_{1:t}, u_{1:t-1}, m_{t-1})$, is estimated using a particle filter which incrementally processes the observations and the odometry readings. Following Doucet *et al.* [18], we can use the motion model $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$ as proposal distribution $\pi(x_t)$ to obtain a sample of the robot pose. This recursive sampling schema allows us to recursively compute the importance weights using the following equation

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{p(z_t \mid x_t^{(i)}, z_{1:t-1}, m_{t-1}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})} \\ &= w_{t-1}^{(i)} p(z_t \mid x_t^{(i)}, z_{1:t-1}, m_{t-1}). \end{aligned} \quad (3)$$

The observation likelihood is then computed by marginalization over the predicted state of the map leading to

$$\begin{aligned} & p(z_t \mid x_t^{(i)}, z_{1:t-1}, m_{t-1}) \\ &= \int p(z_t \mid x_t^{(i)}, m_t) p(m_t \mid m_{t-1}^{(i)}) dm_t \\ &= \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \Sigma), \end{aligned} \quad (4)$$

were z^j is an observation of an occupied point and $\hat{z}_t^j$ is the its closest cell in the map with an occupancy probability above a certain threshold. The map motion model $p(m_t \mid m_{t-1}^{(i)})$ is computed using the HMM and described in section III. Note that the *disappearance* of the integral is not an approximation but a direct consequence of using the likelihood field model [19].

A. Map Management

As we already mentioned above, the initial pose of the robot is unknown and assumed to be uniformly distributed over the environment. This forces us to use a high number of particles, generally above thousands, to accurately represent the initial distribution. Since every particle needs to have its own estimate of the map, memory management is a key aspect of the whole algorithm. It is worth to notice that even if memory is becoming cheap and available in large quantity, the amount needed is still beyond what is currently available. As an example, in an environment of $200\text{x}200\,\text{m}^2$, stored at a resolution of $0.1\,\text{m}$, and using 10000 particles we need about $50\,\text{GB}$ of memory. In order to save memory, we want to only store the cells in the map that have been considerably changed from the a priori map m_0, which is shared among the diverse particles. This is done by exploiting two important aspects of the Markov chain associated to the HMM: the *stationary distribution* and the *mixing time*.

As the number of time steps for that no observation is available tends to infinity, the occupancy value of a cell converges to a unique stationary distribution π [20]. This stationary distribution represents the case where the environment has not been observed for a long time and represents our a priori map m_0. In the case of a binary HMM as the one used in this paper, this distribution is computed using the transition probabilities

$$\begin{bmatrix} \pi_f \\ \pi_o \end{bmatrix} = \frac{1}{p+q} \begin{bmatrix} q \\ p \end{bmatrix}, \quad (5)$$

where for notation simplicity we have

$$\begin{aligned} p &= p(c_t = o \mid c_{t-1} = f), \\ q &= p(c_t = f \mid c_{t-1} = o). \end{aligned}$$

Every time an individual particle observes the state of a cell for the first time, the state distribution of that particular cell changes from the stationary one and the particle needs to store the new state of the cell. In order to reduce memory requirements, only a limited number of cells should be stored and a *forgetting* mechanism should be implemented. This can be done in a sound probabilistic way, by exploiting the mixing

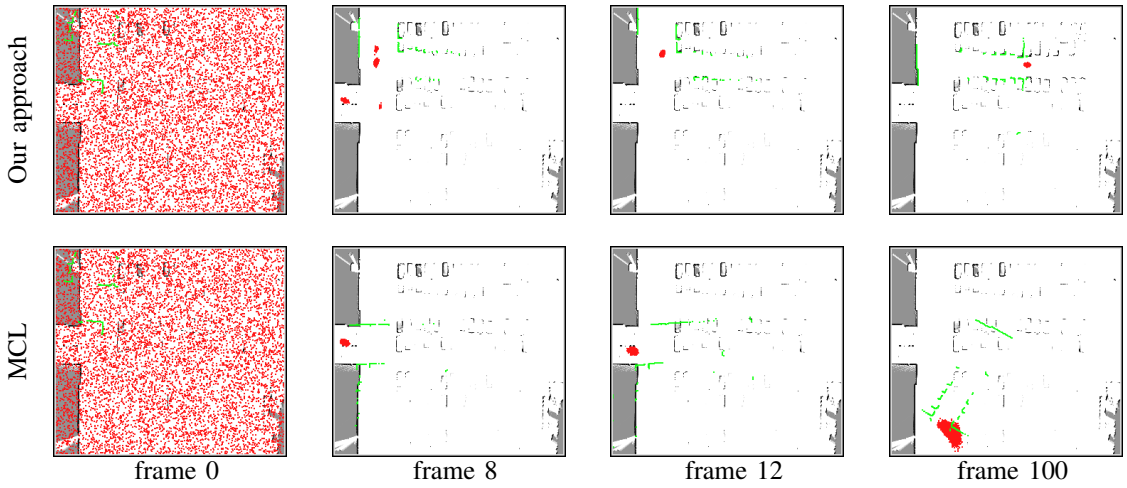


Fig. 2. Comparison between the proposed approach (top) and MCL (bottom) in a global localization setting. The MCL converges too fast and to a wrong position (frame 8), while the proposed approach needs more time to better estimate the current configuration (frame 12). The last frame shows the updated map with the current configuration.

time of the associated Markov chain. The mixing time is defined as the time needed to converge from a particular state to the stationary distribution. The concrete definition depends on the measure used to compute the difference between distributions. In this paper we use the total variation distance as defined by Levin *et al.* [20]. Since our HMMs have only two states, the total variation distance Δ_t between the stationary distribution π and the occupancy distribution p_t at time t can be specified as

$$\Delta_t \quad = \quad |1-p-q|^t \Delta_0, \tag{6}$$

where $\Delta_0 = |p(c_t = f) - \pi_f| = |p(c_t = o) - \pi_o|$ is the difference between the current state $p(c_t)$ and stationary distribution π. Based on the total variation distance, we can define the mixing time t_m as the smallest t such that the distance Δ_{t_m} is less than a given confidence value ϵ. This leads to

$$t_m = \left\lceil \frac{ln(\epsilon/\Delta_0)}{ln(|1-p-q|)} \right\rceil. \tag{7}$$

In other words, the mixing time tells us how many steps are needed for a particular cell to return to its stationary distribution, given an approximation error of ϵ, i.e., how many steps a particle needs to store an unobserved cell before removing it from its local map and rely on the a priori map m_0.

It is worth to notice that the map management reduces the computational complexity as well, since in the naive approach every cell has to be updated in the prediction step, while in our case we need to update only the cells belonging to the local maps.

V. Experiments

We tested our proposed approach using a data set collected with a MobileRobots Powerbot equipped with a SICK LMS laser range finder. In the data set, the robot has been steered through a general parking lot, performing a run every hour from 7am until 6pm during one day. The range data obtained from the twelve runs (data sets d_1 through d_{12}) corresponds to twelve different configurations of the parked cars, including an almost empty parking lot (data set d_1) and a relatively occupied one (data set d_8). A standard SLAM approach for static environments [17] has been used to correct the odometry of the robot in each data set and thereby obtain a good estimate of its pose to use as ground truth. The underlying assumption here is that the parking lot did not change considerably during a run.

In order to assess the performances of the localization approach, we compared it to standard Monte Carlo localization both in a global localization and pose tracking setting. Furthermore, we compared our approach with the one of Meyer-Delius *et al.* [13] but only in the pose tracking case, since it employs standard MCL before the filter reaches convergence. Comparison with other approaches is not possible since they assume either a fixed number of configurations [12] or focused on a limited set of dynamic objects (e.g., doors) [10, 11]. For each data set, we compared our approach (RBPF), MCL using the standard occupancy grid (MCL-S), MCL using the ground-truth map for that specific data set (MCL-GT), and MCL using the temporary maps (MCL-TM). We performed 100 runs for each data set, where we randomly sampled the initial pose of the robot. In order to obtain a fair comparison, the same seed has been used to generate the initial pose, as well as to perform all the random sampling processes for each approach. All the approaches have been initialized with $10,000$ particles for global localization and 500 particles for pose tracking. They also used the same set of parameters, an occupancy threshold of 0.6 and a maximum distance of 1m for the likelihood field model.

For MCL-GT we computed a separate map for each dataset and use it to localize the robot. Since the parking lot did not change during this time, this respect the assumptions of MCL. For MCL-S we stacked all the dataset together and computed

TABLE I
GLOBAL LOCALIZATION EXPERIMENT

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

a map as if the robot run for the whole day. For the RBPF, we learned the parameters of the HMM using n-fold cross validation and used the limiting distribution as prior map m_0.

The results of the global localization experiment are shown in Table I. The table shows the success rate of the global localization, as the percentage of time the filter converged to the true pose, and the residual squared error, with respective variance, after convergence. The success rate is reported relative to the result of MCL on the ground-truth map, in order to have a measure independent of the complexity of the environment. The results show that our approach outperforms the standard MCL on static maps both in terms of convergence rate and accuracy in localization.

Table II shows the results for the pose tracking experiment, where the filter is initialized around the true pose and keeps tracking the robot. The table shows the failure rate, i.e., the percentage of time the robot got lost during tracking, as well as the residual squared error in the case the tracking was succesful. The results of this experiment show that the performance of our approach in pose tracking is almost equivalent to MCL with the ground-truth maps, with a failure rate of only 2%. It is worth to notice the comparison with the temporary map approach [13] in this experiments. Looking at the tables, we get two important messages. The first message is that the proposed approach is always more precise in terms of residual error. This come with no surprise, since the estimation of the local surrounding is in some sense constraint with the map geometry and static objects can only appear in places where they have been seen during learning. On the contrary, the dynamic maps get initialized with the current pose estimate of the robot and introduce a bias in the estimation which is almost impossible to remove. The second message is that the proposed approach is more robust to the changes and initialization. This is evident from the failure rate, where the temporary maps approach is almost always on par with the RBPF but in three cases, and in one case is even worse than standard MCL. The problem is that if the temporary map is created from a wrong position, there is no possibility to recover, the worst case being when the observations matching the prior map are considered as outliers.

In terms of runtime and size of the local map, we experienced and average mixing time of $k = 10$ and an average size of the local map of about 250 cells. The original map size is 369x456 pixels with a resolution of 0.1 m, resulting in a memory saving of about three orders of magnitude with respect to the naive RBPF. A frame to frame comparison between the proposed approach and standard MCL is shown in Figure 2.

Both experiments show two important aspects of the problem and of the solution adopted. The first aspect is that the problem is much more complex than global localization since the search space is bigger and deciding if a measurement is an outlier or is caused by a change of the configuration is not a trivial task. Furthermore, analyzing the performance results in pose tracking, we see that if the filter is initialized close to the correct solution, i.e., the search is reduced to the correct subspace, it is able to estimate the correct configuration. The second aspect is how the algorithm scales with different amounts of change in the environment configuration. In the first four data sets, the parking lot is almost empty and it becomes quite full in the last ones. This is evident, when analyzing the results of MCL on the static maps, since the performance gets worse with an increasing amount of change. On the other hand, the performance of our approach is less sensible to the amount of change in case of global localization and is even independent from that in case of position tracking, as can be seen from the two tables.

VI. CONCLUSIONS

In this paper, we presented a probabilistic localization framework which recursively estimates not only the pose of the robot, but also the state of the environment. Our algorithm uses a hidden Markov model to represent the dynamics of the environment and employs a Rao-Blackwellized particle filter to efficiently estimate the joint state. It explicitly exploits the properties of Markov chains to reduce the memory requirements so that it can be run online on a real robot. Our approach has the main advantages that it provides robust localization even in changing environments and that it additionally supplies

TABLE II
POSITION TRACKING EXPERIMENT

Data set	MCL-GT			RBPF			MCL-S			MCL-TM		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07	0%	0.25	0.16
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10	0%	0.16	0.04
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04	12%	0.63	0.45
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02	29%	0.63	0.51
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02	1%	0.51	0.31
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12	0%	0.21	0.05
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01	1%	0.44	0.24
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15	35%	0.59	0.56
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16	4%	0.49	0.31
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10	0%	0.32	0.12
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05	1%	0.47	0.28
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20	0%	0.23	0.04
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08	7%	0.41	0.25

up-to-date maps of the environment. We evaluated it using real-world data. The results demonstrate that our model significantly outperforms standard Monte-Carlo localization on static maps. This makes our method more suitable for long-term operation of mobile robots in changing environments.

In future, we would like to extend our model to reason about objects and not only about individual cells and experimenting with different models to encode the change (e.g. Dynamic Bayesian Networks and second order Hidden Markov Models). This will bring a novel perspective on how to reason about correlations in a grid map, as well as interestins issues such as moving object detection and motion segmentation.

REFERENCES

[1] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[2] D. Schulz, D. Fox, and J. Hightower, "People tracking with anonymous and id-sensors using rao-blackwellised particle filters," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003.

[3] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[4] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Grid-based models for dynamic environments," Dept. of Computer Science, University of Freiburg, Tech. Rep., 2011.

[5] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[6] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *Int. J. Rob. Res.*, 2007.

[7] L. Montesano, J. Minguez, and L. Montano, "Modeling the static and the dynamic parts of the environment to improve sensor-based navigation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[8] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[9] K. Murphy, "Bayesian map learning in dynamic environments," in *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, Denver, CO, USA, 1999, pp. 1015–1021.

[10] D. Avots, E. Lim, R. Thibaux, and S. Thrun, "A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[11] A. Petrovskaya and A. Y. Ng, "Probabilistic mobile manipulation in dynamic environments, with application to opening doors," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

[12] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, 2005.

[13] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard, "Temporary maps for robust localization in semi-static environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[14] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[15] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1985.

[16] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition." in *Proceedings of the IEEE*, vol. 77 (2), 1989, pp. 257–286.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[18] A. Doucet, J. de Freitas, K. Murphy, and S. Russel, "Rao-Blackwellized partcile filtering for dynamic bayesian networks," in *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, USA, 2000, pp. 176–183.

[19] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[20] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

Simultaneous Localization and Dynamic State Estimation in Reconfigurable Environments

AIS Autonomous Intelligent Systems

Gian Diego Tipaldi Daniel Meyer-Delius
Maximilian Beinhofer Wolfram Burgard

KUKA

Autonomous Intelligent Systems Lab

University of Freiburg, Germany

KUKA Laboratories GmbH

Autonomous Navigation

- Robot must:
  - Know its position in the world and where to go
  - Have a common reference frame with the user
  - Avoid obstacles on the way

- In "robotic" words:
  - Map building
  - Global localization
  - Obstacle avoidance

Autonomous Navigation

- Robot must:
  - Know its position in the world and where to go
  - Have a common reference frame with the user
  - Avoid obstacles on the way

- In "robotic" words:
  - Map building
  - Global localization
  - Obstacle avoidance

Typical Assumption

- The environment does not change over time

- Typical approach
  - Drive the robot around to collect data
  - Use the data to build a map of the environment
  - Localize the robot using that map

- Problem: environments DO change
  - Robot get confused: localization fails
  - Map is out of date: path planning fails

Difficulties of real environments

- Environments change over time
- Different kind of change:
  - Fast changes: people, cars, bikes
  - Mid-Term changes: parked cars, blockages
  - Long-Term changes: new buildings

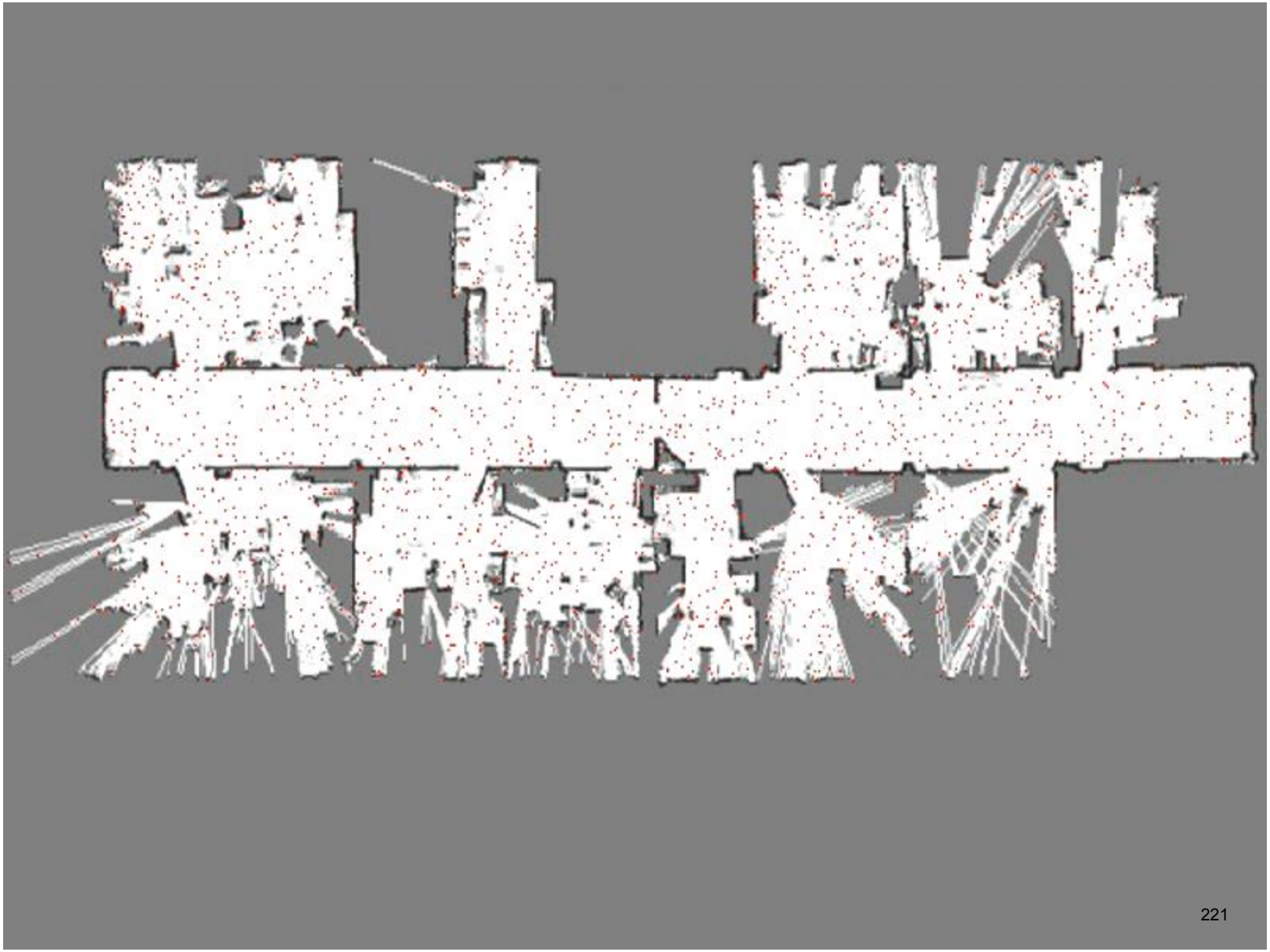
- Properties of changes
  - Fast: happens very often, similar to outliers
  - Mid-Term: happens often, stay for a while
  - Long-Term: happens rarely, structural change

Difficulties of real environments

- Environments change over time
- Different kind of change:
  - Fast changes: people, cars, bikes
  - Mid-Term changes: parked cars, blockages
  - Long-Term changes: new buildings

- Properties of changes
  - Fast: happens very often, similar to outliers
  - Mid-Term: happens often, stay for a while
  - Long-Term: happens rarely, structural change

Naïve 1 - Filter Dynamic Objects



Naïve 1 – Analysis

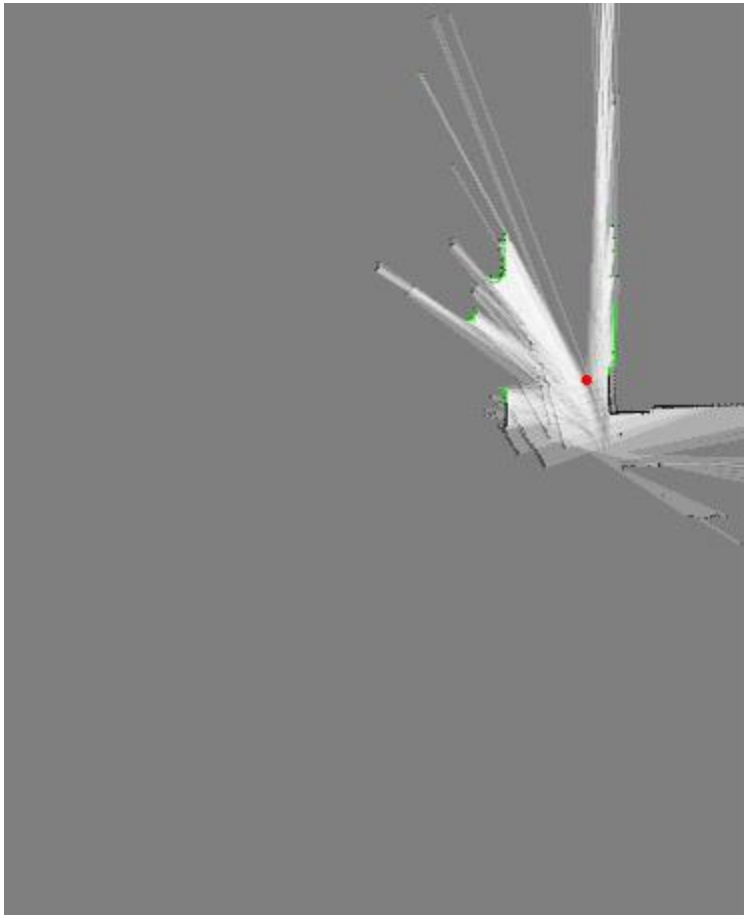
Advantages

- Efficient
- Global reference
- Fast changes

Disadvantages

- Mid-Term changes
- No map update

Naïve 2 – Infinite SLAM



Naïve 2 – Analysis

Advantages

- Map update (slow)
- Mid-Term changes

Disadvantages

- Fast changes
- Complex
- Local reference

Naïve Solutions - Comparison

Dynamic Filtering

Advantages

- Efficient
- Global reference
- Fast changes

Disadvantages

- Mid-Term changes
- No map update

Infinite SLAM

Advantages

- Map update (slow)
- Mid-Term changes

Disadvantages

- Fast changes
- Complexity
- Local reference

Naïve Solutions - Comparison

Dynamic Filtering

Advantages

- Efficient
- Global reference
- Fast changes

Disadvantages

- Mid-Term changes
- No map update

Infinite SLAM

Advantages

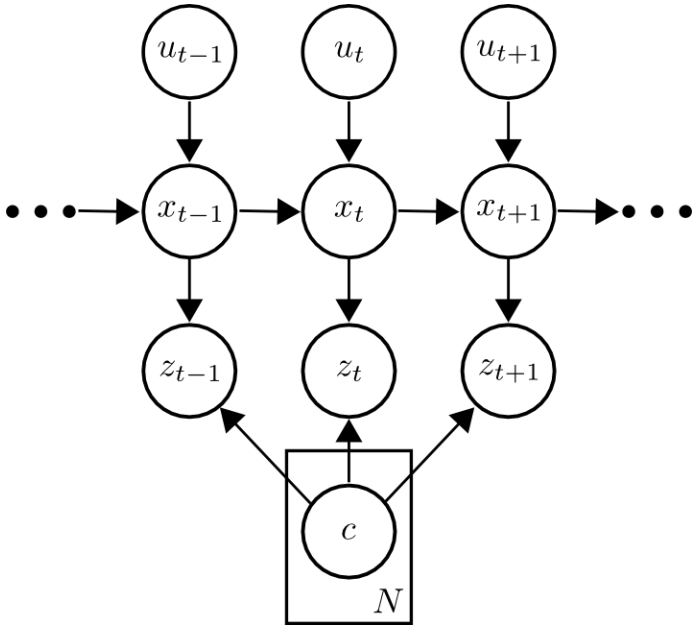
- Map update (slow)
- Mid-Term changes

Disadvantages

- Fast changes
- Complexity
- Local reference

Localization and Map Estimation

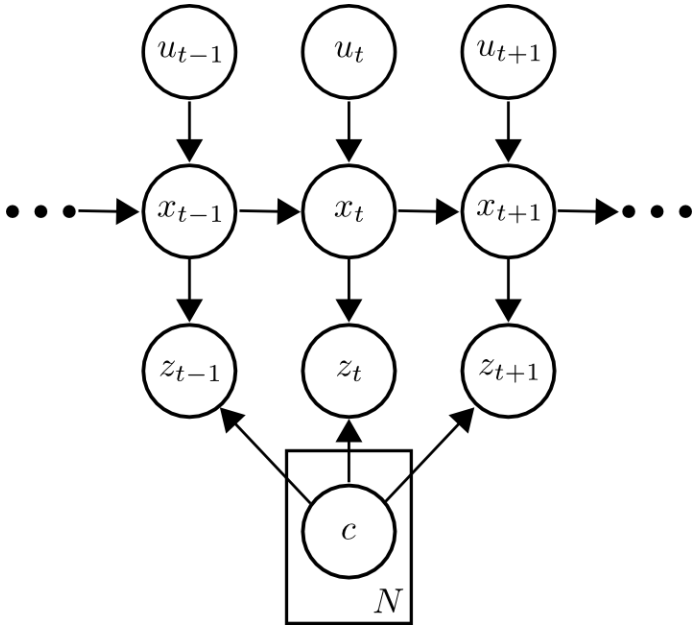
- Idea:
  - Simultaneous Localization and Mapping
  - Reuse the general map structure
  - Model the rate of change of the world



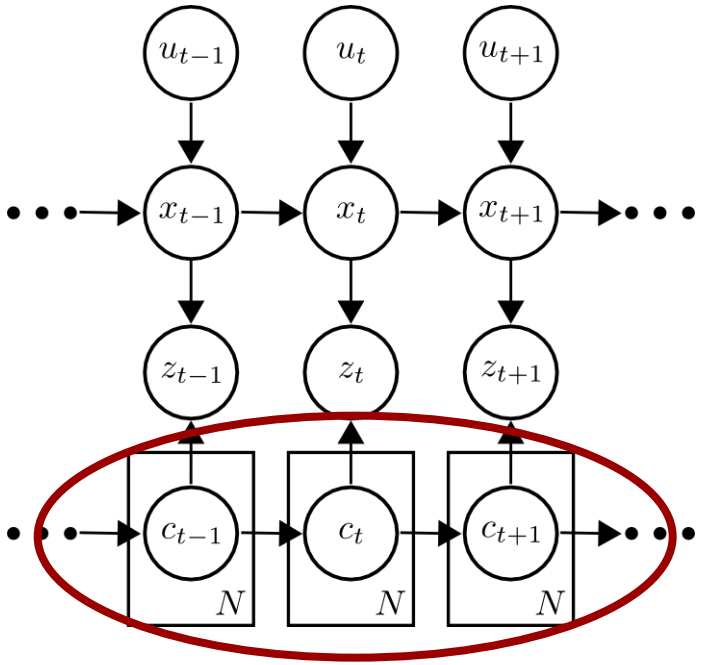
Global localization

Localization and Map Estimation

- Idea:
  - Simultaneous Localization and Mapping
  - Reuse the general map structure
  - Model the rate of change of the world



Global localization



Our approach

Dynamic Occupancy Grid

- HMM represents (for each cell in the grid)
  - Belief about occupancy state, and
  - state transition probabilities

- Specified by
  - Initial state distribution
  - Observation model
  - Transition model

$$p(c_0)$$

$$p(z_t \mid c_t)$$

$$p(c_t \mid c_{t-1})$$

Localization and Map Estimation

- Exploit the same factorization of SLAM

$$p(x_{1:t}, m_{1:t} \mid z_{1:t}, u_{1:t-1}, m_0) = p(m_{1:t} \mid x_{1:t}, z_{1:t}, m_0) p(x_{1:t} \mid z_{1:t}, u_{1:t-1}, m_0).$$

Map estimation **Localization**

- Use Rao Blackwellized Particle filters
  - Multimodal distribution
  - Uniform prior over pose belief space

Rao Blackwellized Particle Filter

Algorithm 1: RBPF for Changing Environments

$\mathcal{S}_t = \{\}$
foreach $s_{t-1}^{(i)}$ *in* $\mathcal{S}_{t-1}$ **do**
  $< x_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} >= s_{t-1}^{(i)}$
  $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$
  foreach c_{t-1} *in* $m_{t-1}^{(i)}$ **do**
    $p(c_t \mid z_{1:t-1}) = \sum_{c_{t-1} \in \{f,o\}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid z_{1:t-1})$
  end
  $w_t^{(i)} = w_{t-1}^{(i)} \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \sigma^2)$
  foreach c_t *in* $m_t^{(i)}$ **do**
    $p(c_t \mid z_{1:t}) = \eta p(z_t \mid c_t) p(c_t \mid z_{1:t-1})$
  end
  $\mathcal{S}_t = \mathcal{S}_t \cup \{< x_t^{(i)}, m_t^{(i)}, w_t^{(i)} >\}$
end
$N_{eff} = \frac{1}{\sum_i (w_t^{(i)})^2}$
if $N_{eff} < T$ **then**
  $\mathcal{S}_t = \text{resample}(\mathcal{S}_t)$
end

- Each particle stores
  - Robot trajectory
  - Map of the environment
- In global localization
  - High number of particles
  - Multimodal distribution

Rao Blackwellized Particle Filter

Algorithm 1: RBPF for Changing Environments

1 $\mathcal{S}_t = \{\}$
2 **foreach** $s_{t-1}^{(i)}$ *in* $\mathcal{S}_{t-1}$ **do**
3 $< x_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} >= s_{t-1}^{(i)}$
4 $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$
5 **foreach** c_{t-1} *in* $m_{t-1}^{(i)}$ **do**
6 $p(c_t \mid z_{1:t-1}) = \sum_{c_{t-1} \in \{f,o\}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid z_{1:t-1})$
7 **end**
8 $w_t^{(i)} = w_{t-1}^{(i)} \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \sigma^2)$
9 **foreach** c_t *in* $m_t^{(i)}$ **do**
10 $p(c_t \mid z_{1:t}) = \eta p(z_t \mid c_t) p(c_t \mid z_{1:t-1})$
11 **end**
12 $\mathcal{S}_t = \mathcal{S}_t \cup \{< x_t^{(i)}, m_t^{(i)}, w_t^{(i)} >\}$
13 **end**
14 $N_{eff} = \frac{1}{\sum_i (w_t^{(i)})^2}$
15 **if** $N_{eff} < T$ **then**
16 $\mathcal{S}_t = \text{resample}(\mathcal{S}_t)$
17 **end**

- Each particle stores
  - Robot trajectory
  - Map of the environment
- In global localization
  - High number of particles
  - Multimodal distribution
- Problem
  - Memory consumption

Rao Blackwellized Particle Filter

Algorithm 1: RBPF for Changing Environments

$\mathcal{S}_t = \{\}$
foreach $s_{t-1}^{(i)}$ *in* $\mathcal{S}_{t-1}$ **do**
  $< x_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} >= s_{t-1}^{(i)}$
  $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$
  foreach c_{t-1} *in* $m_{t-1}^{(i)}$ **do**
    $p(c_t \mid z_{1:t-1}) = \sum_{c_{t-1} \in \{f,o\}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid z_{1:t-1})$
  end
  $w_t^{(i)} = w_{t-1}^{(i)} \prod_j \mathcal{N}(z_t^j; \hat{z}_t^j, \sigma^2)$
  foreach c_t *in* $m_t^{(i)}$ **do**
    $p(c_t \mid z_{1:t}) = \eta p(z_t \mid c_t) p(c_t \mid z_{1:t-1})$
  end
  $\mathcal{S}_t = \mathcal{S}_t \cup \{< x_t^{(i)}, m_t^{(i)}, w_t^{(i)} >\}$
end
$N_{eff} = \frac{1}{\sum_i (w_t^{(i)})^2}$
if $N_{eff} < T$ **then**
  $\mathcal{S}_t = \text{resample}(\mathcal{S}_t)$
end

- Each particle stores
  - Robot trajectory
  - Map of the environment
- In global localization
  - High number of particles
  - Multimodal distribution
- Problem
  - Memory consumption
- Solution
  - HMMs help us!

Stable Distribution & Mixing Time

Stable distribution

- Fixed point solution

$$\pi = A\pi$$

$$\begin{bmatrix} \pi_f \\ \pi_o \end{bmatrix} = \frac{1}{p+q} \begin{bmatrix} q \\ p \end{bmatrix}$$

- with

$$p = p(c_t = o \mid c_{t-1} = f)$$
$$q = p(c_t = f \mid c_{t-1} = o)$$

Mixing time

- Total variation distance

$$\Delta_t = |1-p-q|^t \Delta_0$$
$$\Delta_0 = |p(f) - \pi_f| = |p(o) - \pi_o|$$

- We have

$$t_m = \left\lceil \frac{ln(\epsilon/\Delta_0)}{ln(|1-p-q|)} \right\rceil$$

Efficient Map Management

- If not observed, a cell converges to its stable distribution
- The speed of convergences is given by the mixing time

- **IDEA**: Only store cells if they differ from the stable distribution
  - Share the stable map
  - Store new cell if observed
  - Forget cell if mixing time is less than 1

Experiments – How to evaluate

- What is the evaluation metric?
- What is the baseline?
- How to set up the evaluation?

Experiments – How to evaluate

- What is the evaluation metric?

  Localization accuracy and reliability

- What is the baseline?

  Localization in static environments

- How to set up the evaluation?

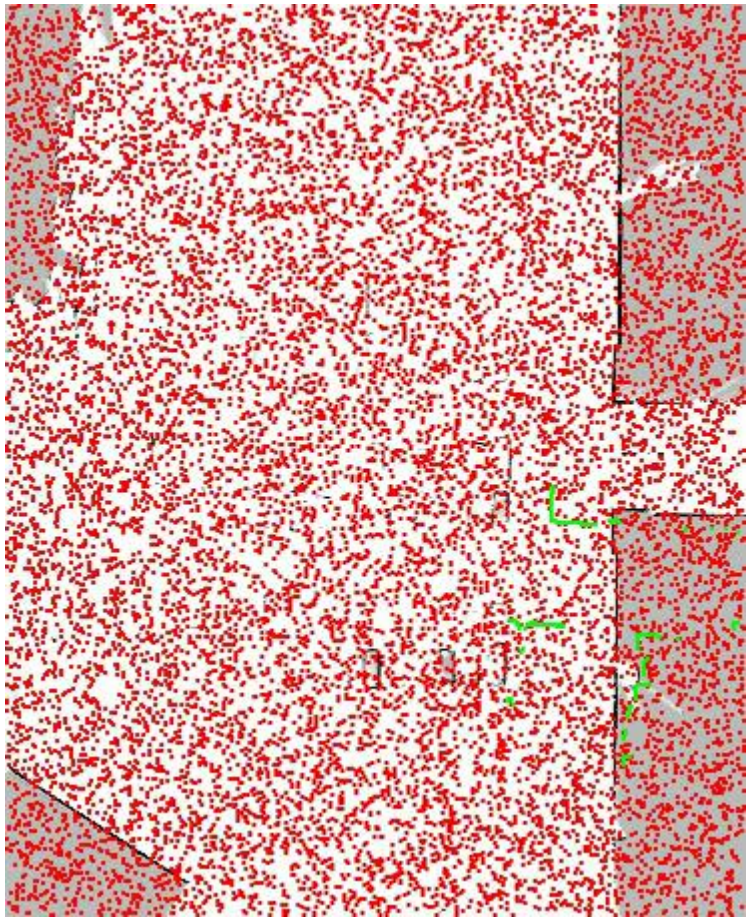
  Static time-slices of dynamic worlds

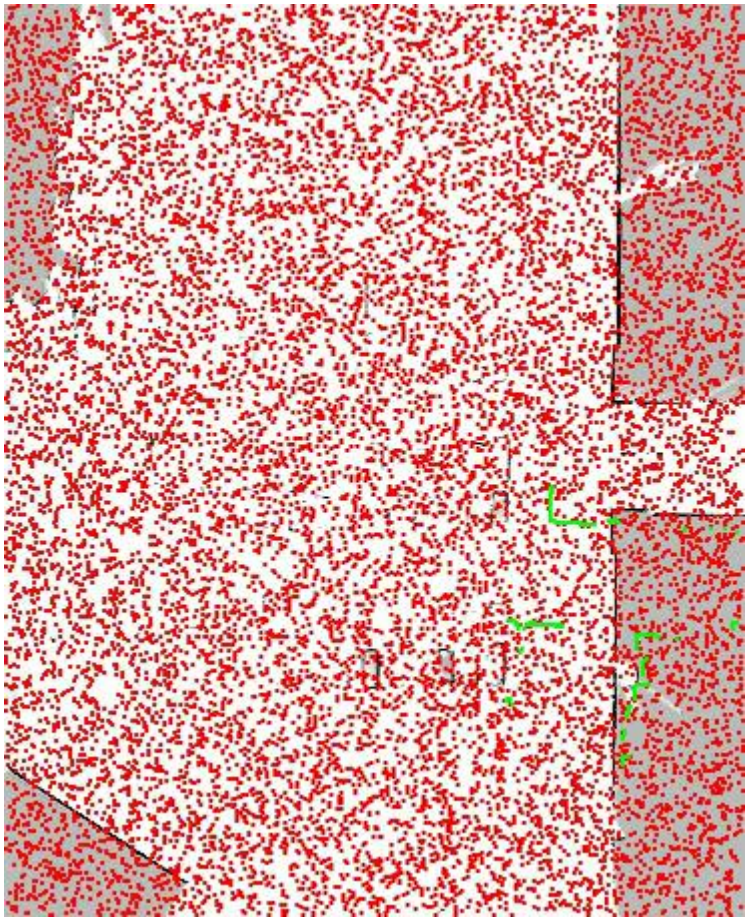
Experiments – Set up

- Parking lot of university of Freiburg
- Data collected every hour (12 logfiles)
- Procedure
  - Compute the "static" map using SLAM
  - Groundtruth using MCL on the "static" map
- Compare with MCL on the overall occupancy grid

Experiment – Global Localization

- Monte Carlo localization
- Our approach





Experiment – Global Localization

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

Experiment – Global Localization

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

Experiment – Global Localization

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

Experiment – Global Localization

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

Experiment – Global Localization

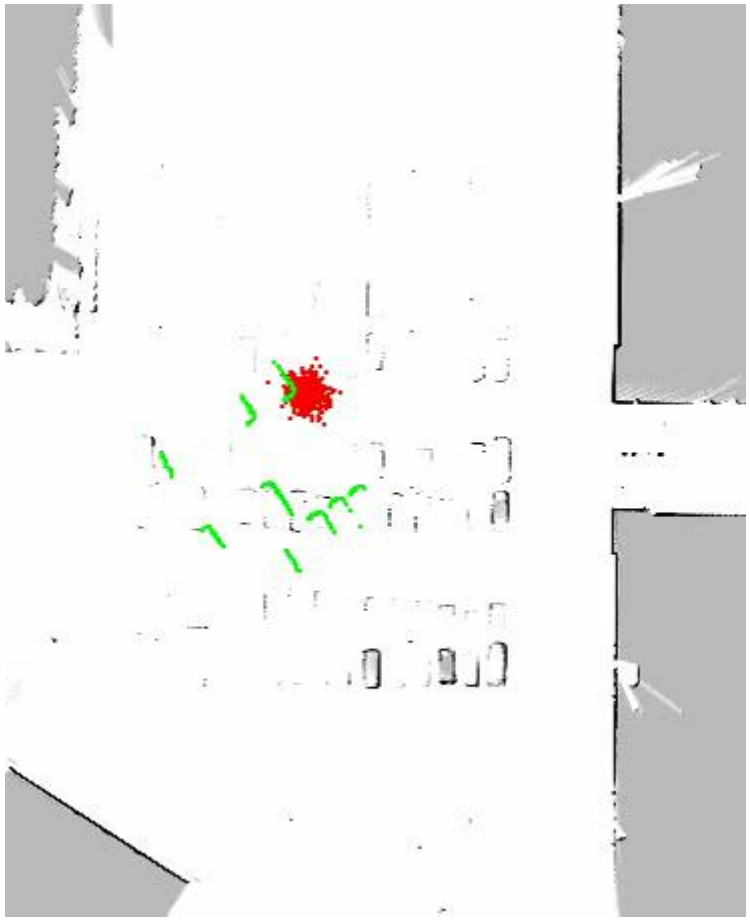
Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

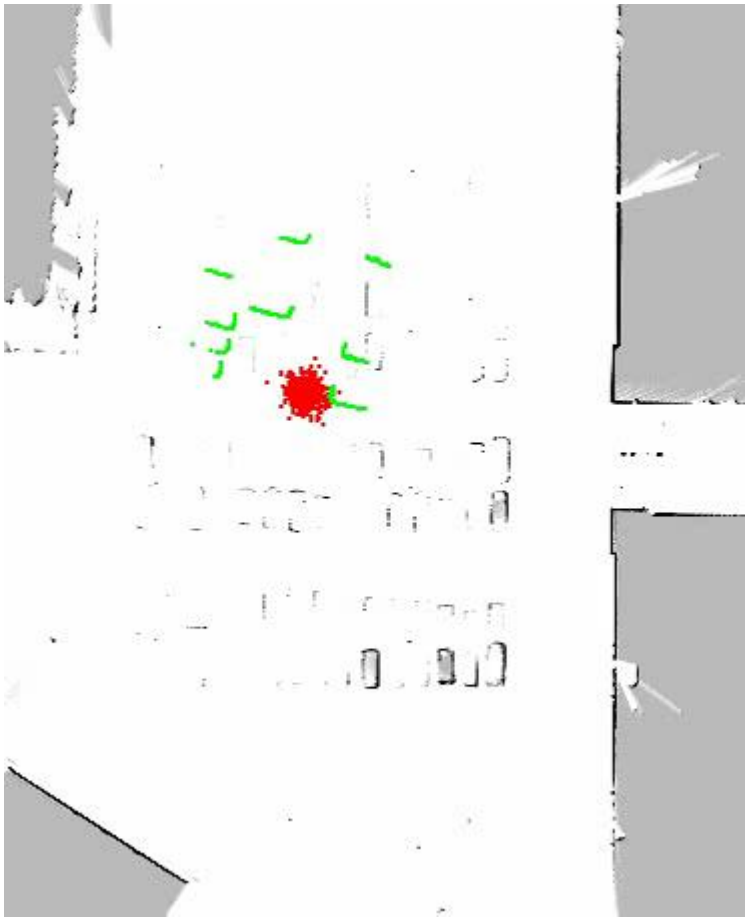
Experiment – Global Localization

Data set	MCL-GT			RBPF			MCL-S		
	Success	Error^2	σ^2	Success	Error^2	σ^2	Success	Error^2	σ^2
01	100%	0.21	0.36	50%	0.26	0.36	50%	0.26	0.18
02	100%	0.19	0.29	40%	0.10	0.08	33%	0.13	0.09
03	100%	0.13	0.19	80%	0.10	0.29	52%	0.19	0.17
04	100%	0.04	0.03	60%	0.08	0.14	53%	0.15	0.19
05	100%	0.07	0.18	54%	0.07	0.09	35%	0.15	0.18
06	100%	0.02	0.01	87%	0.02	0.02	45%	0.06	0.02
07	100%	0.06	0.08	59%	0.12	0.22	43%	0.14	0.20
08	100%	0.05	0.10	71%	0.03	0.02	28%	0.03	0.01
09	100%	0.02	0.01	53%	0.12	0.22	31%	0.06	0.02
10	100%	0.14	0.28	62%	0.13	0.31	34%	0.30	1.01
11	100%	0.11	0.11	38%	0.15	0.21	26%	0.24	0.29
12	100%	0.19	0.32	20%	0.16	0.14	22%	0.27	0.38
Total	100%	0.11	0.19	52%	0.11	0.18	36%	0.17	0.22

Experiment – Pose Tracking

- Monte Carlo localization
- Our approach





Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

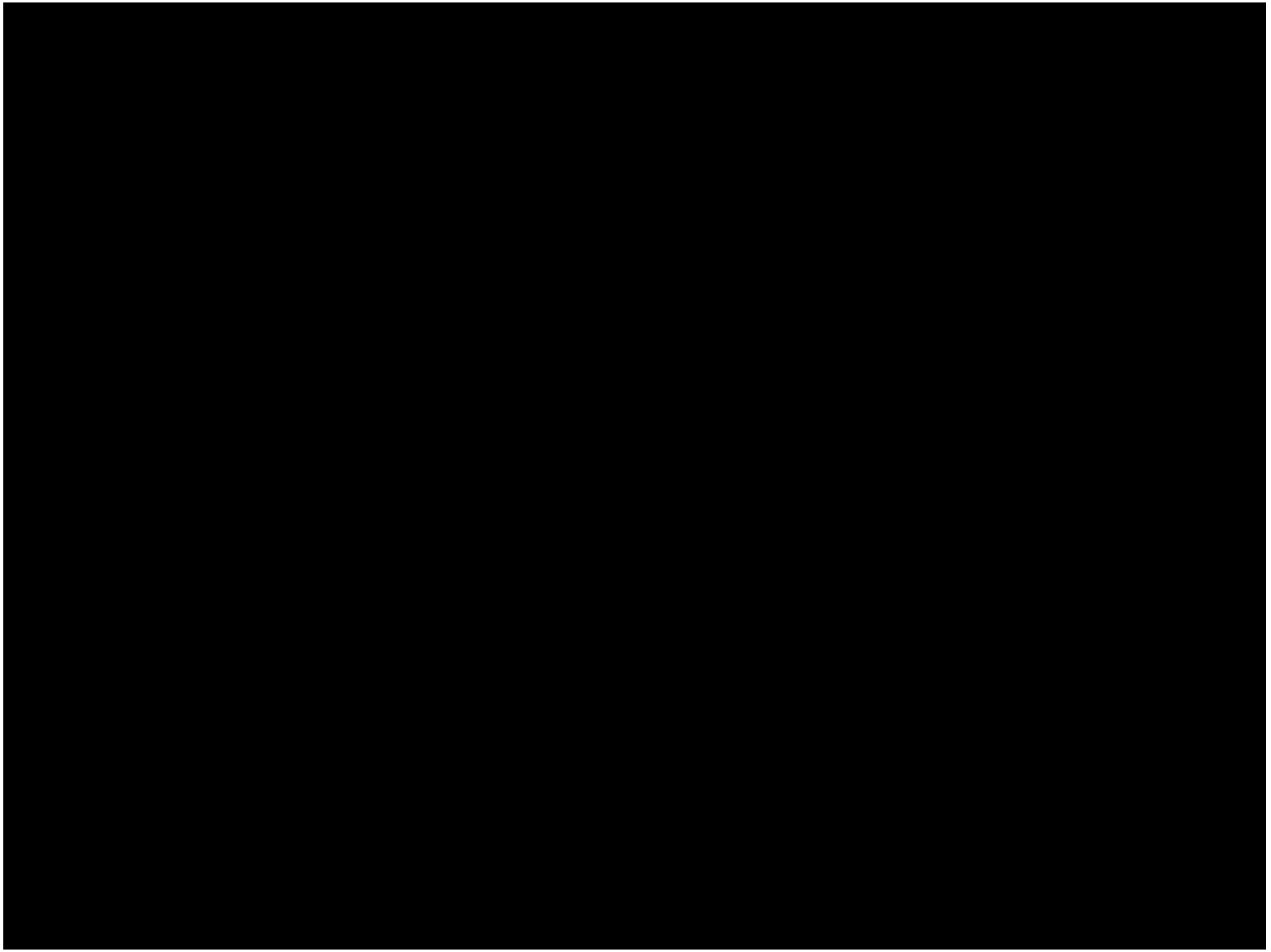
Experiments – Position Tracking

Data set	MCL-GT			RBPF			MCL-S		
	Failure	Error^2	σ^2	Failure	Error^2	σ^2	Failure	Error^2	σ^2
01	0%	0.04	0.01	3%	0.09	0.03	5%	0.18	0.07
02	0%	0.03	0.01	4%	0.08	0.05	24%	0.18	0.10
03	0%	0.04	0.01	2%	0.05	0.04	10%	0.09	0.04
04	0%	0.02	0.01	0%	0.04	0.01	10%	0.08	0.02
05	0%	0.02	0.01	3%	0.03	0.04	13%	0.06	0.02
06	0%	0.02	0.01	2%	0.02	0.01	26%	0.09	0.12
07	0%	0.02	0.01	0%	0.03	0.01	34%	0.07	0.01
08	0%	0.02	0.01	2%	0.02	0.01	35%	0.09	0.15
09	0%	0.02	0.01	4%	0.03	0.01	37%	0.07	0.16
10	0%	0.02	0.01	0%	0.03	0.01	36%	0.09	0.10
11	0%	0.03	0.01	1%	0.05	0.02	42%	0.10	0.05
12	0%	0.03	0.01	5%	0.06	0.01	44%	0.15	0.20
Total	0%	0.03	0.01	2%	0.04	0.02	27%	0.10	0.08

Conclusions

- Work with **KUKA** for the EU Project **TAPAS**
- Novel localization solution in dynamic environments
- Directly model the dynamics
- Efficient solution using HMMs
  - Stable distribution
  - Mixing times
- Experiments with real robot shows improved stability and accuracy

Video



2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Session V

Perception & Situation awareness

- **Title: Real-time Scan-Matching Using L0-norm Minimization Under Dynamic Crowded Environment**
  Authors: Y. Hieida, T. Suenaga, K. Takemura, J. Takamatsu and T. Ogasawara

- **Title: Fast classification of static and dynamic environment for Bayesian Occupancy Filter (BOF)**
  Authors: Q. Baig, M. Perrollaz, J. Botelho Do Nascimento, C. Laugier

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7th, 2012 Vilamoura, Algarve, Portugal

Real-time Scan-Matching Using L_0-norm Minimization Under Dynamic Crowded Environments

Yusuke Hieida[1], Tsuyoshi Suenaga[1], Kentaro Takemura[1], Jun Takamatsu[1] and Tsukasa Ogasawara[1]

***Abstract*— We propose real-time scan-matching based on L_0-norm minimization under dynamic crowded environment. The prior scan-matching methods are based on L_2-norm minimization, because the measurement noise follows the normal distribution in static environments. This assumption is unfortunately broken in dynamic crowded environments.**

We propose to use the idea of Locality Sensitive Hashing (LSH) to accelerate the L_0-norm minimization, which usually is a time-consuming process. The LSH customized for our issue reduces the calculation time even in the worst cases. The experimental results demonstrate the effectiveness of the proposed method compared with standard L_2-norm minimization and its robust version with M-estimator.

I. Introduction

It is very demanded for an autonomous robot to move under unknown environments [1]. *Scan-matching* estimates robot's displacement by aligning time-series measurements. Concatenation of displacement and the aligned measurement generate robot's trajectory and the map respectively, while may suffer from accumulation of alignment errors. Iterative Closest Point (ICP) method [2] and particle filter [3], [4] improve the performance under static environments [5].

However, the assumption valid for static environments is broken in real-world environments. Under dynamic crowded environments, the accuracy of the alignment deteriorates because both static (referred to as *inliers*) and moving objects (referred to as *outliers*) are rigidly aligned, resulting in poor performance of the prior methods (Fig. 1). To robustify outliers in the alignment is inevitable under dynamic environments.

A. Related work

There are two types of approaches for dynamic environments. One approach explicitly detects inliers/outliers by feature extraction, tracking and so on. The other approach gradually or simultaneously estimates the likelihood of outliers and decreases their effect during the alignment.

In the first approaches, Wolf and Sukhatme [6] proposed to use only landmarks, which tend to be static, as inliers. Examples of landmarks are wall planes and corners. Wang *et al*. [7] propose to interleave object tracking based on motion detection. Hähnel *et al*. [8] also propose to use object tracking based on the sample joint probabilistic data association filter. Rodriguez-Losada and Minguez [9] prove that the possible displacement to satisfy correspondence of one point between two frames is helix and detect the outliers by using the distance to the helix. The performance of these methods deeply depends on the detection of inliers/outliers.

In the second approaches, Hähnel *et al*. propose to use the expectation-maximization (EM) algorithm [10]. The E-step evaluates likelihood of outliers and the M-step aligns measurements with weighting according to the likelihood. Ramos *et al*. [11] propose to use the conditional random field (CRF) to stochastically estimate data association and outliers. Further, Ven *et al*. [12] estimate moving object and motion as well as data association. Since these methods need to optimize the function with large number of parameters, it is difficult to achieve real-time operation.

Viewing from the outlier detection, use of robust statistics, such as M-estimator [13], is one of the solutions. There are several methods that use M-estimator for scan matching-based alignment [14], [15]. Although the optimization of the M-estimator is simpler and thus faster than the second approaches, it may also lack of inappropriate convergence.

B. Proposed method

We propose real-time scan-matching that suits adapted to dynamic environments. We use a common real-time LIDAR sensor, which measures depths only on the cross-sectional plane. Thus, we assume that the ground is flat and the measuring cross-section is parallel to the ground. We believe that the proposed method can be extended to its 3-D version.

The novelty of the proposed system is to use L_0-norm being that the prior systems use L_2-norm. Although optimization of the L_0-norm tends to be time-consuming, we propose to use the Locality Sensitive Hashing (LSH) [16] to make the proposed method suitable for real-time.

The Contributions of this paper are twofold:

- We experimentally prove that the effectiveness of the L_0-norm for scan-matching under dynamic environments. Note that such norm is often employed in other applications, such as face recognition [17], image inpainting [18], and denoising [19].
- We propose to accelerate the L_0-norm calculation with approximation using the LSH [16]. Unlike the L_2-norm calculation, we only need to decide if the other points exist inside the pre-defined radius and it is not necessary to consider their point ID. These aspects stabilize the calculation time even in the worst cases and reduce memory consumption by removing the point ID information.

[1]They are with Graduate School of Information Science, Nara Institute of Science and Technology, Japan {yusuke-h, tsuyo-s, kenta-ta, j-taka, ogasawar} at is.naist.jp

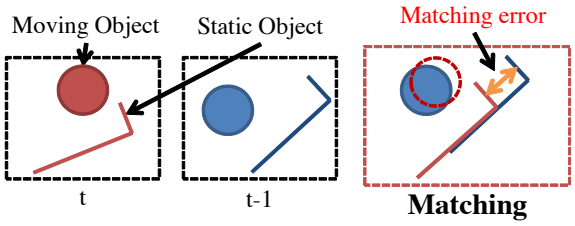


Fig. 1. Erroneous estimation of displacement due to dynamic environment. The circular object moves rightward from time $t-1$ (middle) to time t (left), while the L-shape object is fixed. Alignment only using the L-shape object outputs the correct displacement. But the prior scan-matching methods diffuse matching errors uniformly in the space, resulting in erroneously estimating the displacement.

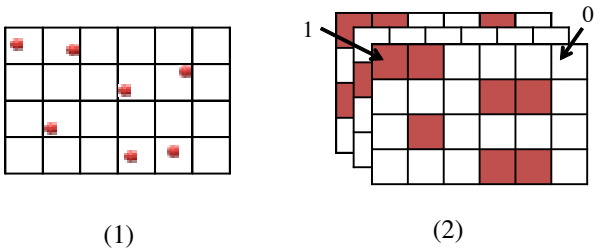


Fig. 2. Several boolean-array tables are generated by the LSH. Each bin represents the existence of r-near neighbors and thus the existence is decided by accessing the corresponding bin in $O(1)$ time.

II. Preliminary

Consider two point sets, $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_j\}$, which are measurements of the same environment, but they have different coordinates, *i.e.*, measurements from different positions. Robot's displacement is estimated by aligning these two sets.

In scan-matching scenario, displacement is estimated by minimizing the evaluation function $E(\mathbf{R},\mathbf{t})$ in Eq. (1) [1].

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{n} f(\mathbf{R}\mathbf{p}_i + \mathbf{t}, \{\mathbf{q}_j\}) = \sum_{i=1}^{n} f(\mathbf{p}'_i, \{\mathbf{q}_j\}), \quad (1)$$

where n is the number of the points $\{\mathbf{p}_i\}$, $\mathbf{p}'_i \stackrel{\text{def}}{=} \mathbf{R}\mathbf{p}_i + \mathbf{t}$, a 2×2 matrix $\mathbf{R}$ represents the orientation and a 2-D vector $\mathbf{t}$ represents the translation in robot's planer displacement. The function f defines the distance between the point $\mathbf{p}'_i$ and the measurement $\{\mathbf{q}_j\}$. In prior methods [2], the function f is Euclidean distance between $\mathbf{p}'_i$ and the closest point $\mathbf{q}_c$ of the measurement, as $f(\mathbf{p}'_i, \{\mathbf{q}_j\}) = ||\mathbf{p}'_i - \mathbf{q}_c||^2$.

III. Scan-matching based on L_0-norm

A. Definition of L_0-norm

Given two sets of points, $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_j\}$, the evaluation function $E(\mathbf{R},\mathbf{t})$ based on L_0-norm is similarly defined as Eq. (1), but the definition of the function f is different, such as

$$f(\mathbf{p}'_i, \{\mathbf{q}_j\}) = \begin{cases} 0 & (\exists j, ||\mathbf{p}'_i - \mathbf{q}_j|| \le \epsilon) \\ 1 & (\text{otherwise}) \end{cases}. \quad (2)$$

The distance threshold ϵ is a constant real number, which is decided based on the accuracy of the measurement. The evaluation function counts the points which do not have any points near the distance less than ϵ. Note that calculation of the L_0-norm does not require the explicit point correspondences, just to decide if the other points exist inside the radius ϵ. We refer to these points as *r-near neighbors* following the terminology in [16].

B. Acceleration by Locality Sensitive Hashing

We employ the Locality Sensitive Hashing (LSH) [16], whose calculation time is $O(1)$. The LSH limits the search space using the hash function, whose values of two proximity points tend to be the same. In the LSH, only the points with similar hash values are considered. Using multiple hash functions reduces the tendency to failure in the search. Concretely, the hash function is defined as

$$h(\mathbf{x}) = \left\lfloor \frac{\mathbf{a}\cdot\mathbf{x} + b}{w} \right\rfloor, \quad (3)$$

where the symbol $\lfloor\ \rfloor$ means floor operation, $\mathbf{x}$ is the data point, w is the width of the hash bin and the parameters $\mathbf{a}$ and b are randomly chosen by p-degree stable distribution and uniform distribution from 0 to w, respectively. We only consider the projection of x- and y-axes as the vector $\mathbf{a}$, *e.g.*, $(a_x, 0)$ and $(0, a_y)$, and use them by concatenating x- and y-axis projection LSH as shown in Fig. 2. The existence of r-near neighbor is searched by accessing the bin of the table several times (at most, number of tables).

Although the LSH is usually used for nearest neighbor search, the LSH in r-near neighbor search has two more advantages. In the nearest neighbor search, when all the searched points $\{\mathbf{q}_j\}$ have different hash values, it is necessary to continue searching the points whose values are near to the target value. However in r-near neighbor search, the calculation is just terminated and we conclude that no r-near neighbor exists. L_0-norm does not consider the position or ID of the r-near neighbor, and depends only on the existence.

IV. Minimization by importance sampling

The approximate robot's displacement is obtained from odometry or control inputs. Thus, we use the sampling-based optimization method considering importance of searched region. In strict sense, minimization of L_0 norm is equivalent to combinatorial optimization. In this sense, using the method in [20] is alternative and future work.

We minimize the evaluation function using importance sampling and coarse-to-fine search as shown in Fig. 3:

1) randomly generate n_1 locations (referred to as sample-A) based on odometry or control inputs
2) calculate the importance of each sample by Eq. (4).
$$\exp\left(\frac{m - E(R,\mathbf{t})}{k}\right). \quad (4)$$
The parameter k controls the degree of the importance with respect to the evaluation function.
3) select n_2 locations (referred to as sample-B) from sample-A following their weights.

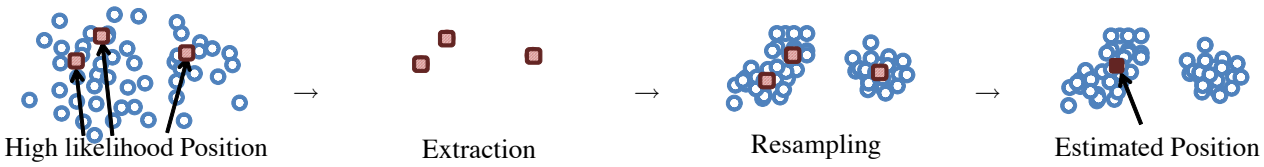


Fig. 3. Minimization by importance sampling and coarse-to-fine search. By gradually changing the distribution σ of sampling range, the samples converge on the minimum in the L_0-norm.

(a) LIDAR: SICK LMS100 (b) Mobile robot: EMC-230

Fig. 4. Equipments used in this experiment

TABLE I
SPECIFICATION OF SICK LMS100

Viewing angle	270 [deg]
Angular resolution	0.5 [deg]
# of points in each frame	540
Accuracy in depth	1.2 [cm]
Maximum depth	20 [m]
Frequency	30 [Hz]

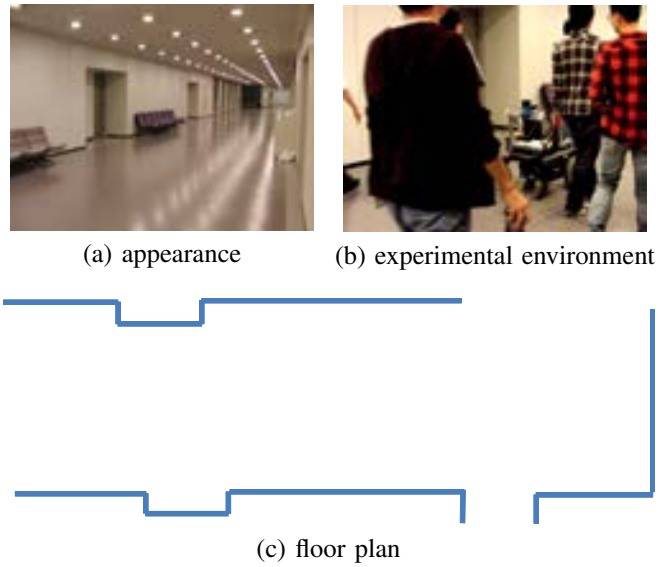
4) randomly generate n_3 locations by normal distribution with variance $(\sigma_x, \sigma_y, \sigma_\theta)$ around from sample-B. $n_2 \times n_3$ locations are sampled in total.
5) repeat 2) to 4) in several times, while reducing variances.

V. EXPERIMENTS

A. Experimental setup

We used the LIDAR, SICK *LMS100* (Fig. 4 (a)), and a wheelchair, *EMC-230* made by Imasen Engineering Corporation as the mobile robot (Fig. 4 (b)). This LIDAR measures uniformly along the angle direction and thus the measured points become sparser as the depth increases. Considering the calculation time and the estimation accuracy, the measured points $\{\mathbf{p}_i\}$ are segmented and thinned out so that the points are distributed uniformly in the space. On the other hand, in the measured points $\{\mathbf{q}_j\}$, additional points are inserted as all the intervals of neighbors are less than the threshold ϵ in Eq. (2). We set ϵ to 1 [cm] based on the accuracy of the LIDAR used in this experiment (see Table I).

We conducted three types of experiments. The first experiment verifies the effectiveness of the L_0-norm in outliers compared with simple ICP, ICP with thresholding, and ICP with M-estimator, which are most related to the proposed method. Note that first approaches described in Section I-A use such methods after detecting outliers. The second experiment verifies the effectiveness of the LSH with respect to the calculation time and accuracy. The third experiment verifies the applicability of the proposed method through actual use. The experimental environment is mainly the inside of a building of our campus.



(a) appearance (b) experimental environment

(c) floor plan

Fig. 5. Experimental environment. (a) appearance of the inside of the building. (b) actual experimental environment crowded by people. (c) floor plan of the environment.

B. Verification of effectiveness of L_0-norm

Figure 5 shows the experimental environment which is about 5 [m] in width and 20 [m] in depth. About 25 people are moving toward the robot within the environment. The robot measures the environment while fixed. Thus, the estimated robot displacement (x, y, θ) should be always zero, where x, y, and θ are the translation along the x- and y-axes and the orientation, respectively.

To verify the effectiveness without considering the initial guess, we optimize all of the functions by brute-force search, where the ranges of x, y, and θ are $[-5\ \text{[cm]}, 5\ \text{[cm]})$, $[-5\ \text{[cm]}, 5\ \text{[cm]})$, and $[-0.3\ \text{[rad]}, 0.3\ \text{[rad]})$ and sampled uniformly at 0.5 [cm], 0.5 [cm], and 0.01 [rad] ($\fallingdotseq$ 0.57 [deg]), respectively. There are 24000 samples in total.

In ICP with thresholding, we ignore some ratio of the further corresponding data as outliers. We employ the Cauchy and the Biweight function as M-estimator $\rho(d)$ and use $\rho(f(\mathbf{R}\mathbf{p}_i+\mathbf{t}, \{\mathbf{q}_j\}))$ in place of $f(\mathbf{R}\mathbf{p}_i+\mathbf{t}, \{\mathbf{q}_j\})$ in Eq. (1). The Cauchy function is defined as Eq. (5) and is often used (*e.g.* [14], [15]):

$$\rho(d) = \frac{C^2}{2} \log\left(1 + \left(\frac{d}{C}\right)^2\right), \tag{5}$$

TABLE II

MSE OF THE ESTIMATED LOCALIZATION: L_2-NORM (SIMILE ICP), THRESHOLD (IGNORE SOME RATIO OF THE FURTHER CORRESPONDING DATA AS OUTLIERS), M-ESTIMATOR (CAUCHY AND BIWEIGHT FUNCTIONS) AND L_0-NORM

Method	MSE (x [cm], y [cm], ϕ [rad])
L_2-norm	(3.37, 2.08, 0.016)
Threshold (30 percents)	(1.17, 0.033, 0)
Threshold (50 percents)	(0.19, 0.004, 0)
Cauchy	(1.33, 0.013, 0)
Biweight	(0.13, 0.002, 0)
L_0-norm	(0.42, 0.015, 0)

TABLE III

COMPARISON AMONG THREE KINDS OF r-NEAR NEIGHBOR SEARCH: SIMPLE METHOD, KD-TREE METHOD, AND LSH. THE LSH OPERATES AT HIGH SPEED, BUT SACRIFICES THE ACCURACY.

method	time [s]	# of localizations per second	RMSE of the L_0-norm
simple	126	190	-
kd-tree	24	1000	-
LSH	5.4	4436	14.4

where C is a positive real value. The Biweight function is defined as Eq. (6) and completely removes the effect of the outliers (shape of the function is flat in the region $|e| \geq 0.01$) similar to the proposed L_0-norm.

$$\rho(d) = \begin{cases} \frac{B^2}{2} & (d \geq B) \\ \frac{B^2}{2}\left(1 - \left(1 - \left(\frac{d}{B}\right)^2\right)^3\right) & (d < B) \end{cases}, \quad (6)$$

where B is a positive real value. Both values, B and C, are set to 1 [cm] considering the sensor's accuracy.

Since the estimated displacement should be zero as described before, we quantitatively compare them using the mean square error (MSE) of the displacement. Table II shows the results. The estimation by the L_0-norm, L_2-norm with higher threshold, and the Biweight function are similar and better than the others, because sensor accuracy is 1.2 [cm]. This result demonstrates the effectiveness of the L_0-norm, which regards the point outside of the pre-defined radius as outliers. The Biweight function and ICP with higher threshold have a similar characteristic and thus achieves a better performance. But this means that these three methods have the same disadvantages with respect to the convergence. Note that distance used as threshold in ICP with thresholding is decided in each brute-force step.

Figures 6 to 9 show the maps estimated by these methods, respectively. Each map is constructed by overlaying all the measurements according to the estimated displacement. The red line shows robot's displacement, which should not appear. The outer boundary shows the static walls of the building (See Fig. 5 (c)), while the inner points indicate moving people. If the map is appropriately estimated, the outer boundary should be aligned without blurring. These figures also demonstrate the effectiveness of the L_0-norm.

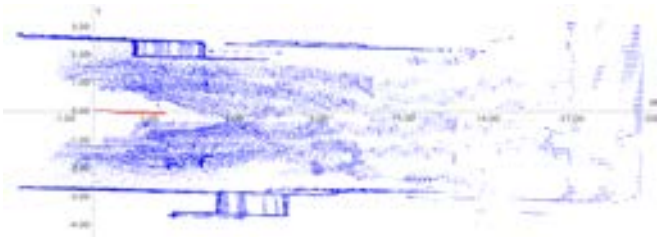


Fig. 6. The map estimated by the L_2-norm minimization with 30 percent threshold. The red line indicates robot's displacement and the blue points indicate measured points on the global coordinates. Since the robot is fixed, the red line should not appear. Since the blue points on the outer boundary corresponds to the wall of the building (Fig. 5 (c)), they should not be blurred.

Fig. 7. The map estimated by the Cauchy function

C. Verification of effectiveness of LSH

We compare the proposed LSH acceleration with the simple method and the kd-tree method [22]. The computer to employ is composed of Intel *Core2Duo T7500* (Clock speed: 2.2 [GHz]) and 2 [GB] memory. We conducted this experiment with the same configuration of the previous experiment, but the robot moved.

Table III shows the average calculation time for the brute-force search (24000 times of calculation), number of L_0-norm calculations per second and the root mean square error (RMSE) of the calculated L_0-norm; the simple and kd-tree methods always output correct answers, but the LSH method does not. In calculation time, the LSH method is about five times faster than the kd-tree method, but sacrifices the accuracy; RMSE of the L_0-norm is 14.4, while the average of the norms is 280.03.

To evaluate the degeneration caused by the approximation of the L_0 norm calculation, we actually construct the map using the LSH method. Figure 10 shows the map which the RBPF-SLAM method [21] estimates from measurement of the static environment and we regard it as ground truth. Figure 11 shows the map which the LSH method estimates from measurement of the same but dynamic environment. Despite the approximation, the estimated map is acceptable in visual inspection.

D. Verification in Actual Use

We actually estimate the map using the proposed method, which includes the LSH method and minimization by the importance sampling. The localization at each frame is performed at about 5 [fps] on the same computer as that of Section V-C. The parameters are: n_1 = 300, n_2 = 20, n_3 = 15, k = 20.0. Steps 2) to 4) are repeated twice,

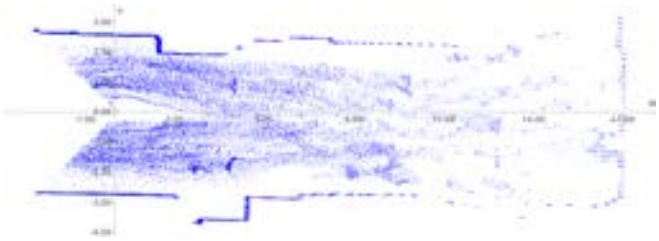


Fig. 8. The map estimated by the Biweight function

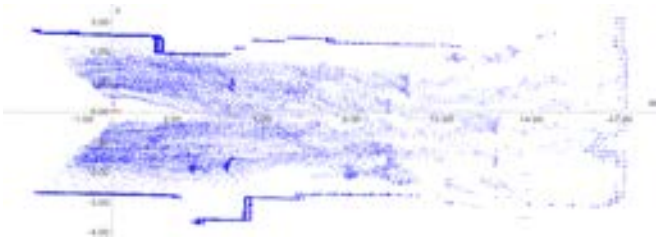


Fig. 9. The map estimated by the L_0-norm minimization

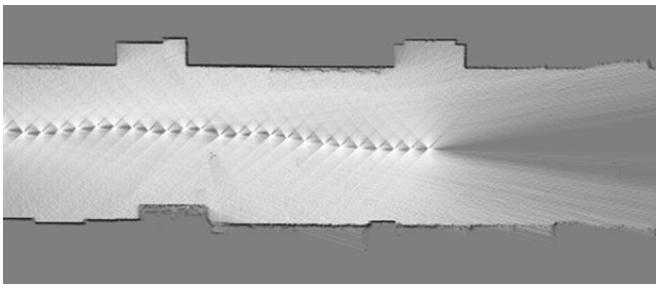


Fig. 10. The map estimated by the RBPF-SLAM method [21] under the environment without moving objects.

Fig. 11. The map estimated by L_0-norm under the same environment as Fig. 10, but including moving objects.

and $(\sigma_x, \sigma_y, \sigma_\theta) = (0.03, 0.03, 0.025)$ at the first time and $(\sigma_x, \sigma_y, \sigma_\theta) = (0.01, 0.01, 0.01)$ at the second time.

In the first experiment, the size of the experimental environment is about 50 [m] $\times$ 30 [m]. About 25 people are moving around the robot as outliers as shown in Fig. 12. 300 frames of measurements are used. Figure 13 shows the estimated map. Figure 14 shows the occupancy grid map obtained from the estimation results using the open SLAM software *MRPT* [1]. We overlay the floor plan of this building to evaluate the accuracy.

Figure 15 shows the estimated map in the second experiment, which includes both indoor and outdoor measurements. The size of the experimental environment is about 100 [m] $\times$ 60 [m]. 700 frames of measurements are used. To evaluate the accuracy of the map, we overlay the aerial photograph downloaded from the Google maps[2].

E. Discussion

Both the L_0-norm, ICP with higher threshold, and the Biweight function have similar advantages and disadvantages. Unlike the other two methods, by ignoring the distance and ID of the r-near neighbor in the LSH, the L_0-norm calculation is accelerated. Despite ignoring them, the localization accuracy is surprisingly maintained as shown in the experiments. Note that better results obtained by these three methods strongly support the plausibility of using the L_0-norm.

VI. Conclusion

In this paper, we proposed real-time scan-matching for dynamic environments, which works at about 5 [fps] on Intel Core2Duo T7500 (Clock speed: 2.2 [GHz]) with 2 [GB] memory. The proposed method uses L_0-norm, not L_2-norm, for alignment. First, we proved the robustness of the L_0-norm to outliers through actual use.

Next, we proposed to accelerate r-near neighbor search by the LSH technique. Unlike the nearest neighbor search, we only need to determine if r-near neighbors exist and it is not necessary to consider the IDs of the neighbors. These aspects do not affect the calculation time even in the worst cases and reduce memory consumption by removing the point ID information.

The proposed method estimates the displacement only from two subsequent measurements. Generally, scan-matching suffers from accumulation of the estimation errors. *Loop-closing* technique [23] is one of the solutions.

[1] http://www.mrpt.org

[2] http://maps.google.com

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. The MIT Press, 2005.

[2] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[3] M. Isard and A. Blake, "CONDENSATION - conditional density propagation for vision tracking," *Int. J. of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[4] S. Godsill, A. Doucet, and M. West, "Maximum a posteriori sequence estimation using monte carlo particle filters," *Ann. Inst. Statist. Math.*, vol. 53, pp. 82–96, 2001.

[5] S. Thrun, "Simultaneous mapping and localization with sparse extended information filters: Theory and initial results," *Algorithmic Foundations of Robotics V*, vol. 7, pp. 363–380, 2003.

[6] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[7] C. C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2003.

[8] D. Hähnel, D. Schulz, and W. Burgard, "Map building with mobile robots in populated environment," in *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

Fig. 12. Snapshots of images obtained from digital camera (left column) and SICK LMS100 (right column) installed in the robot.

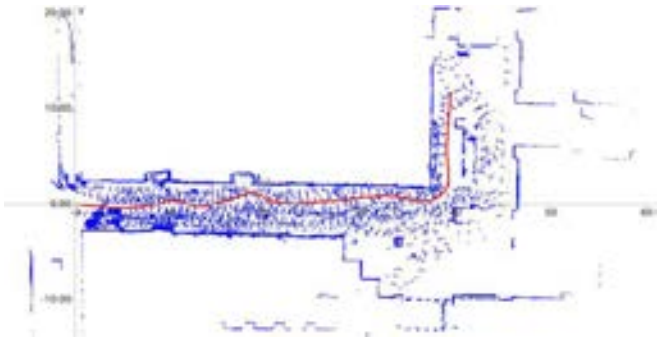


Fig. 13. The map estimated by the L_0-norm minimization

Fig. 14. The occupancy grid map obtained by the open SLAM software *MRPT* and the proposed method. We overlay the floor plan of this building to evaluate the accuracy.

Fig. 15. The map estimated by the proposed method. We overlay the corresponding aerial photograph from the *Google Maps* to evaluate the accuracy.

[9] D. Rodriguez-Losada and J. Minguez, "Improved data association for icp-based scan matching in noisy and dynamic environments," in *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2007.

[10] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with moblile robots in dynamic environments," in *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2003.

[11] F. Ramos, D. Fox, and H. Durrant-Whyte, "Crf-matching: Conditional random fields for feature-based scan matching," in *RSS*, 2007.

[12] J. van de Ven, F. Ramos, and G. D. Tipaldi, "An integrated probabilistic model for scan-matching, moving object detection and motion estimation," in *ICRA*, 2010.

[13] P. J. Huber, *Robust statistics*. Wiley-Interscience, 1981.

[14] M. D. Wheeler and K. Ikeuchi, "Sensor modeling, probabilistic hypothesis generation and robust localization for object recognition," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 17, no. 3, pp. 252–265, 1995.

[15] K. Kawamura, K. Hasegawa, Y. Someya, Y. Sato, and K. Ikeuchi, "Robust localization for 3D object recognition using local EGI and 3D template matching with m-estimators," in *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[16] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, *Locality-sensitive hashing scheme based on p-stable distributions, nearest neighbor methods in learning and vision*. MIT Press, 2006.

[17] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2009.

[18] L. Mancera and J. Portilla, "L0-norm-based sparse representation through alternate projections," in *Proc. of Int. Conf. on Image Processing (ICIP)*, 2006.

[19] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *Proc. of Int. Conf. on Comp. Vis. and Patt. Anal. (CVPR)*, 2010.

[20] E. B. Olson, "Real-time correlative scan matching," in *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[21] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[22] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *ACM Commun.*, vol. 18, pp. 509–517, 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007

[23] C. Stachniss and W. Burgard, "Exploration with active loop-closing for FastSLAM," in *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

Fast classification of static and dynamic environment for Bayesian Occupancy Filter (BOF)

Qadeer Baig, Mathias Perrollaz, Jander Botelho Do Nascimento, Christian Laugier
The authors are with the E-Motion team, Inria Rhone-Alpes,
38334 Montbonnot Saint Martin, France
Email: First.Last@inria.fr

***Abstract*—In this paper we present a fast motion detection technique based on laser data and odometry/imu information. This technique instead of performing a complete SLAM (Simultaneous Localization and Mapping) solution, is based on transferring occupancy information between two consecutive data grids. We plan to use the output of this work for Bayesian Occupancy Filter (BOF) framework to reduce processing time and improve the results of subsequent clustering and tracking algorithm, based on BOF. Experimental results obtained from a real demonstrator vehicle show the effectiveness of our technique.**

I. Introduction

In the field of Advanced Driver Assistance Systems (ADAS), many current approaches rely on the perception of the road scene. Particularly, the detection and tracking of the objects in the scene is essential for prediction of risky driving situations. Among the recent approaches for risk estimation, the authors in [1] propose to model and recognize the behavior of road scene participants. This approach is very promising for long term prediction of the risk, but not applicable for short term evaluation of the scene where we need to separate environment into static and dynamic parts.

A huge amount of work has been done to detect moving objects, especially by vision community [2]–[5]. These techniques have primarily been based on background subtraction or motion cues. Similar to background subtraction techniques have also been used in occupancy grids to detect moving objects [6]–[8]. These techniques are based on inconsistencies observed for new data by comparing them with maps constructed by SLAM. [9] and [10] have also proposed model based techniques to detect moving objects on the roads in the context of autonomous vehicle driving. Recently Dempster-Shafer theory based grids (evidential grids) have been used to detect moving objects using conflict analysis techniques [11], [12].

There exist various approaches for such classification of the environment. Having a very precise map of the environment, coupled with a very accurate localization algorithm, can be costly. Thus to perform a SLAM-based localization would be more realistic for commercial vehicles. The computed occupancy grid provides a description of the static environment [13]. More elaborated techniques like [8] or [9] improve this approach by performing detection and classification simultaneously, in a model-based-tracking framework.

In this paper, we propose a fast and efficient method for static/dynamic environment classification, which can be plugged in Bayesian Occupancy Filter [14], [15], to provide a more accurate representation of cell velocities. This will result in an improved results of Fast Clustering and Tracking Algorithm (FCTA) [16]. Our approach is different from other grid based methods in the sense that usually a complete SLAM solution is implemented to separate the moving parts from the static parts (as in [10]). However in current work we have developed a technique that deals with only two consecutive frames to detect moving parts rapidly. Our claim is that, by removing all static objects from the grid, the performance of the FCTA algorithm would increase effectively.

The paper is organized as follow: In next section we describe our demonstrator vehicle used to get data sets for this work with sensors installed on it. Section III describes the Bayesian occupancy filter framework for which we plan to use the results of current work. Next in section IV we detail our technique to detect moving objects from the sensor data. We present some results in section V and conclude this work with future perspectives in section VI.

II. Demonstrator

Our experimental platform is a Lexus LS600h car shown in Fig. 1. The car is equipped with such sensors as: two IBEO Lux lidars placed in the front bumper, one on left and other on the right of the vehicle, a TYZX stereo camera situated behind the windshield, and an Xsens IMU with GPS. Extrinsic calibration of the sensors is done manually for this work. Note that, thanks to the grid-based approach and considering the resolution of the grid, a slight calibration error has very little impact on the final results.

The hardware specification are the following: IBEO Lux LIDAR laser scanner provides four layers of up to 200 beams with a sampling period of 20 ms. The angular range is $100°$, and the angular resolution is $0.5°$. The on-board computer is equipped with 8GB of RAM, an Intel Xeon 3.4GHz processor and a NVIDIA GeForce GTX 480 for GPU. The observed region is 60 m long by 20 m wide, with a maximum height of 2 m (due to different verticle angle of the 4 layers of each laser scanner). Cell size for the occupancy grids is 0.2x0.2 m. The car is equipped with an IMU sensor: MTi-G XSens which

is responsible for collecting the innertial data, it is deployed in the middle of the rear wheel axis, In this work we do not use the stereo vision cameras.

Different sensors installed on the demonstrator vehicle and other hardware setup are shown in figures 1 , 2 and 3.

Fig. 1. Lexus LS600h car equipped with two IBEO Lux lidars and a TYZX stereo camera

Fig. 2. MTi-G XSens IMU unit

Fig. 3. Intel Xeon 3.4GHz linux box

III. Bayesian Occupancy Filter (BOF)

In this section we briefly introduce the BOF framework with FCTA module to detect and track the moving objects. In BOF framework the idea is to perform sensor fusion and environment monitoring at low level, hence the fast and efficient grid based representation during all the processing. Objects are only retrieved at the end of the processing through clustering of the dynamic parts of the grid. So the complete processing BOF framework is divided into three stages: i) Multi sensor fusion using occupancy grid representation ii) filtering and estimation of dynamic grid and finally iii) Clustering of the objects and tracking. These three parts are elaborated below.

A. Sensor fusion from the multiple lidar layers

Each of the two lidar sensors installed on the vehicle provides 4 layers of scanning points. Each layer is used to compute an occupancy grid using the classical approach described in [17]. In order to retrieve a single grid for representation of the environment, the data from all these layers are merged using the approach described in [18]. This approach fuses the sensory information by using Linear Opinion Pools [19]. It has the advantage of reducing the errors due to conflicting information from the multiple layers.

The principle is to generate a posterior distribution over the occupancy of a cell C of the grid given the opinion of m sensors $\{Y_1 \dots Y_m\}$. Each sensor gives two quantities: its estimation for the occupancy of the cell $P(C|Y_i)$ and $w_i(C)$, a measure of the confidence for such estimations. The idea is to shut-down those sensors that do not give relevant information to the process by assigning a low weight to them. The fusion of all sensory information will be as follows:

$$P(C|Y_1 \dots Y_m) = \alpha \sum_{i=1}^{m} w_i(C) P(C|Y_i) \qquad (1)$$

where $\alpha = \left[\sum_{i=1}^{m} w_i(C) \right]^{-1}$ is a normalization factor for the weights. Equation (1) is used to generate 2D-occupancy grids. For each sensor Y_i we must define $P(C|Y_i)$, the probability of a cell being occupied given the sensor information; and $w_i(C)$, the confidence on the opinion. Note that we assume independence among cells. This assumption, though it is very strong, is necessary to be efficient in computing equation (1), for each cell in parallel.

B. Filtering the grid using the Bayesian Occupancy Filter

The Bayesian Occupancy Filter (BOF) framework provides filtering capability, as well as the ability to estimate a velocity distribution for each cell of the grid. The BOF [14], [15] provides an adaptation of the Bayesian filtering methodology to the occupancy grid framework. It is based on a prediction-estimation paradigm. As an input, it uses an observed occupancy grid. On its output, it provides an estimated occupancy grid and a velocity grid, representing the probability distribution over possible velocities for each cell. An efficient formulation of this filter can be found in [20].

The BOF operates with a four-dimensional grid representing the environment. Each cell of the grid contains a probability distribution of the cell occupancy and a probability distribution

of the cell velocity. Given a set of observations, the BOF algorithm updates the estimates of the occupancy and velocity for each cell in the grid. The inference leads to a Bayesian filtering process, as shown in Fig. 4.

In this context, the prediction step propagates cell occupancy and antecedent (velocity) distributions of each cell in the grid and obtains the prediction $P(O_c^t\ A_c^t)$ where $P(O_c^t)$ denotes the occupancy distribution and $P(A_c^t)$ denotes the antecedent (velocity) distribution of a cell c at time t. In the estimation step, $P(O_c^t\ A_c^t)$ is updated by taking into account the observations yielded by the sensors $\prod_{i=1}^{S} P(Z_i^t \mid A_c^t O_c^t)$ to obtain the a posteriori state estimate $P(O_c^t\ A_c^t \mid [Z_1^t \cdots Z_S^t])$ where Z_i^t denotes the observation of sensor i at t. This allows us to compute by marginalization $P(O_c^t \mid [Z_1^t \cdots Z_S^t])$ and $P(A_c^t \mid [Z_1^t \cdots Z_S^t])$, which will be used for prediction in the next iteration.

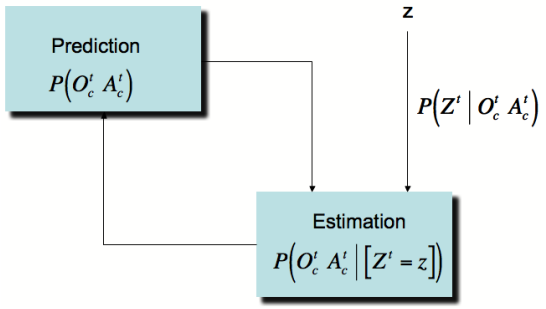


Fig. 4. Bayesian filtering in the estimation of occupancy and velocity distributions in the BOF grid

C. Detecting objects with the Fast Clustering and Tracking Algorithm

Obstacle detection requires to retrieve an object level representation of the scene. This can not be directly reached from the occupancy grid, and therefore a clustering algorithm is necessary. An algorithm adapted to the BOF framework is the "Fast Clustering Tracking Algorithm" described in [16]. It has the major interest to create clusters considering not only the connectivity in the occupancy grid, but also the Mahalanobis distance between cells in the estimated velocity grids. Thus two connected cells with different velocities are not merged during the clustering process.
FCTA includes a Kalman filter for target tracking and a ROI prediction approach that allows computation to be performed in real time. The output of the algorithm is a set of tracked objects, with position, velocity and associated uncertainties.

IV. Motion Detection

In this section we detail the technique that we have developed to find moving parts of the evironment. The input to this motion detection module consists of an occupancy grid generated by the fusion module described in previous section and that fuses data from eight layers of two laser scanners installed on the demonstrator vehicle. Let us represent this occupancy grid at time t as $OG_t[i]$ where $0 \leq i < N$ with N being the total cells of this occupancy grid. The value of each cell of this grid is between 0 and 1 i.e. $0 \leq OG_t[i] \leq 1$ and represents internal belief of the ego vehicle about the occupancy state of each cell with 0 means empty and 1 means occupied.

The output of the *XSens MTi-G* motion sensor installed on the demonstrator, at time instant t, consists of (along with other information) two components of velocity $v_t = (v_x, v_y)$ and values of quaternion components for orientation $Q_t = (q_0, q_1, q_2, q_3)$. From these information we calculate the translational and rotational velocities $u_t = (\nu_t, \omega_t)$ of the demonstrator vehicle as follows.

$$\nu_t = \sqrt{v_x^2 + v_y^2} \tag{2}$$

To calculate rotational velocity of the vehicle we calculate yaw angle of the vehicle from the quaternion as follows

$$\mathcal{Y} = atan2(2*(q0*q3+q1*q2), 1-2*(q2*q2+q3*q3)) \tag{3}$$

And if dt is the time difference between two successive data frames at time t and $t-1$ then rotational speed ω at time t is equal to the yaw rate given as:

$$\omega_t = \frac{\mathcal{Y}_t - \mathcal{Y}_{t-1}}{dt} \tag{4}$$

At each time instant t these OG_t and u_t are input to the algorithm that consists of following steps.

I) Free and occupied counts arrays: for each new input occupancy grid OG_t we create two count arrays, the first one called $FreeCount_t$ and the other called $OccupiedCount_t$ to keep count of the number of times a cell has been observed free and number of times it has been observed occupied respectively. These arrays are initialized from OG_t as follows:

$$OccupiedCount_t[i] = \begin{cases} 1, & \text{if } OG_t[i] > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

and

$$FreeCount_t[i] = \begin{cases} 1, & \text{if } OG_t[i] < 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

II) Counts update from previous step: Suppose $FreeCount_{t-1}$ and $OccupiedCount_{t-1}$ are the updated counts arrays at time $t-1$ and we want to update new counts $FreeCount_t$ and $OccupiedCount_t$ from these old counts. Since vehicle has undergone a position change determined by $u_t = (\nu_t, \omega_t)$, so there is no direct correspondence between cells of two occupancy grids OG_t and OG_{t-1}. We must find transformations that map a cell in OG_{t-1} to a cell in OG_t using u_t. This situation is shown in figure 5, OG_{t-1} has origin at O_{t-1} and OG_t has origin at O_t. To find this transformation suppose $O_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1}) = (0, 0, 0)$ is the pose (position and orientation) of the occupancy grid at time instant $t-1$ (i.e of OG_{t-1}) and we want to find $O_t = (x_t, y_t, \theta_t)$, the pose of OG_t under u_t. Considering a circular motion trajectory, the pose of O_t w.r.t O_{t-1} is given as:

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} \nu_t/\omega_t * sin(\omega_t * dt) \\ \nu_t/\omega_t - \nu_t/\omega_t * cos(\omega_t * dt) \\ \omega_t * dt \end{bmatrix} \quad (7)$$

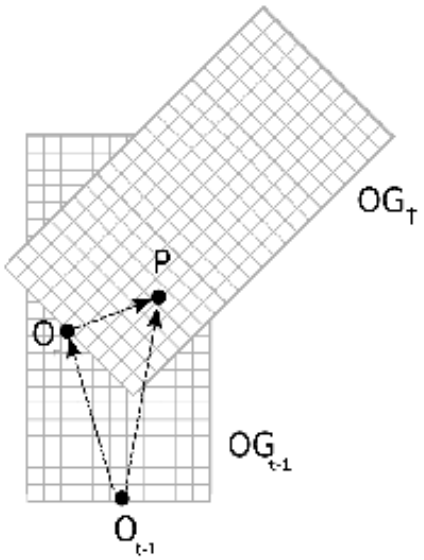


Fig. 5. Positon of the grid at time instants $t-1$ and t. Vehicle undergoes a motion of $u_t = (\nu_t, \omega_t)$ to move from O_{t-1} to O_t. We need to find the position of point P of grid OG_{t-1} in grid OG_t.

An important thing to note here is that we are concerned with the localization of two consecutive frames only. We do not solve the complete SLAM problem, making the technique very fast and avoiding the error accumulation over time. Moreover empirically observed odometry error between two consective frames is less than 10 cm whereas the cell size is 20cm x 20cm enabling us to assume that cell mappping (explained next) from grid at $t-1$ to grid at t is exact.

To map a cell of grid OG_{t-1} to grid OG_t we proceed as follows. Suppose point P (shown in figure 5) is the center of a cell in grid OG_{t-1} and we want to find its corresponding cell in grid OG_t. We define following two pose manipulation operations:

If P_{ij} is the pose of origin j w.r.t origin i and $P_{jk} = [x_{jk}, y_{jk}, \theta_{jk}]^T$ is the pose of origin k w.r.t j then the pose of k w.r.t i denoted as $P_{ik} = [x_{ik}, y_{ik}, \theta_{ik}]^T$ is given as:

$$P_{ik} \equiv \oplus(P_{ij}, P_{jk}) = \begin{bmatrix} x_{jk}\cos(\theta_{ij}) - y_{jk}\sin(\theta_{ij}) + x_{ij} \\ x_{jk}\sin(\theta_{ij}) + y_{jk}\cos(\theta_{ij}) + y_{ij} \\ \theta_{ij} + \theta_{jk} \end{bmatrix} \quad (8)$$

For the pose P_{ij} the reverse pose relationship $P_{ji} = [x_{ji}, y_{ji}, \theta_{ji}]^T$ (pose of i w.r.t j) is defined as:

$$P_{ji} \equiv \ominus(P_{ij}) = \begin{bmatrix} -x_{ij}\cos(\theta_{ij}) - y_{ij}\sin(\theta_{ij}) \\ x_{ij}\sin(\theta_{ij}) - y_{ij}\cos(\theta_{ij}) \\ -\theta_{ij} \end{bmatrix} \quad (9)$$

Since the pose of O_t w.r.t O_{t-1} is $P_{O_{t-1}O_t} = [x_t, y_t, \theta_t]^T$ and point P has pose $P_{O_{t-1}P} = [x, y, 0]^T$ w.r.t O_{t-1}. The pose of this point P w.r.t O_t is calculated as.

$$P_{O_tP} = \oplus(P_{O_tO_{t-1}}, P_{O_{t-1}P}) \quad (10)$$

or

$$P_{O_tP} = \oplus(\ominus(P_{O_{t-1}O_t}), P_{O_{t-1}P}) \quad (11)$$

First two components of P_{O_tP} give x and y positon of point P w.r.t origin O_t. From these x and y values we can easily calculate the index of the cell where point P lies in grid OG_t. These transformations will map a cell having index i in OG_{t-1} to a cell having index j in grid OG_t. If this cell j is visible in OG_t i.e $0 \leq j < N$ then we can update new count values for this cell as follows:

$$FreeCount_t[j] = FreeCount_t[j] + FreeCount_{t-1}[i] \quad (12)$$

and

$$OccupiedCount_t[j] = OccupiedCount_t[j] + OccupiedCount_{t-1}[i] \quad (13)$$

We repeat this process for all cells of grid OG_{t-1} to update counts values in grid OG_t.

III) Motion detection: After the counts arrays have been updated as explained above the motion grid can be calculated from the new data using following heuristic:

$$MotionGrid_t[i] = \begin{cases} 1, & OG_t[i] > 0.5 \text{ and} \\ & FreeCount_t[i] > 2 * OccupiedCount_t[i] \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

After this processing $MotionGrid_t$ has 1s in the cells which are detected as belonging to moving objects.

V. Results

Some qualitative results are shown in figures 6, 7, 8 and 9 (rectangles around the objects are drawn manually to highlight them). Figure 6 shows the motion detection scenario of two cars. A car moving around a round point has been successfully detected in figure 7. Detection of two moving cars on a highway is shown in figure 8. We see some false positives as well but we believe that this noise can be easilyl removed by the later steps. Finally figure 9 shows the case when there is no moving object in the view, we see that no significant object is detected as moving. A video showing some more results of this work can be found here[1].

VI. Conclusion and Future Work

In this paper we have presented a fast technique to find moving objects from laser data. The presented technique does not require to perform complete SLAM to detected moving objects but uses laser data along with odometry/IMU information to transfer occupancy information between two consecutive grids.

We plan to use this fast motion detection technique for the BOF framework to provide a priori motion information for the

[1] https://sites.google.com/site/qadeerbaig/motion-detection

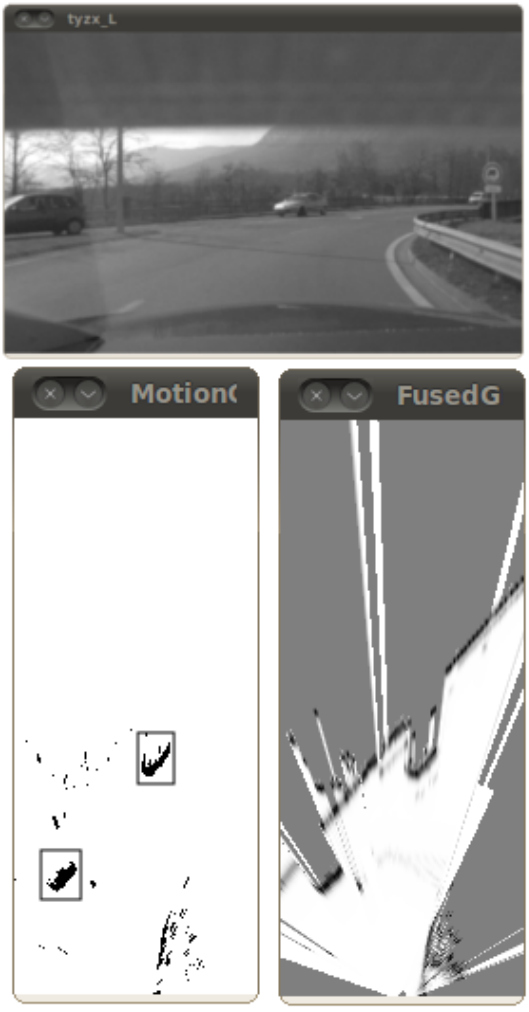


Fig. 6. Motion detection results of two cars. Top, scenario, bottom right input fused grid, bottom left resulting motion grid.

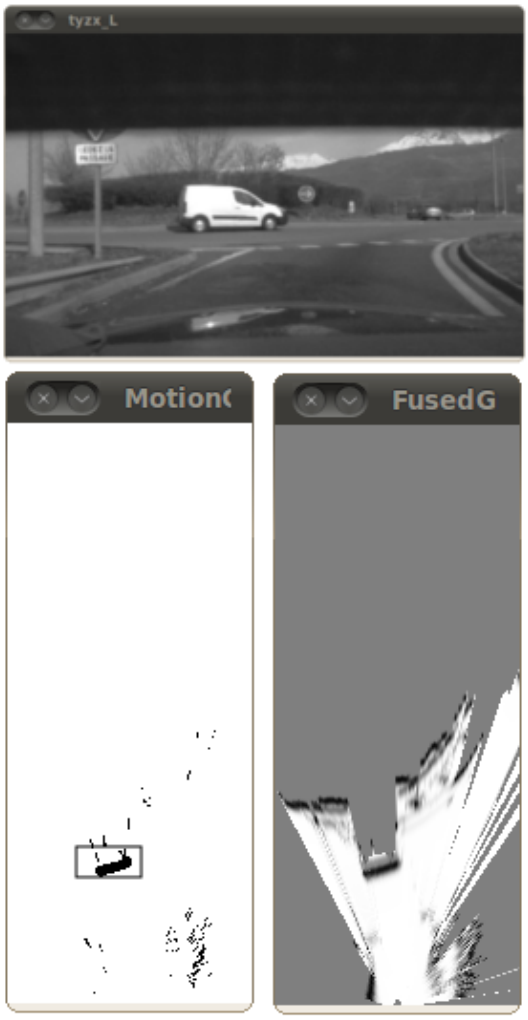


Fig. 7. Motion detection results of a car on a round point. Some noise due to sensor uncertainty is also visible

calculation of cell velocities. Currently BOF, in the absence of a motion sensor, uses a bayesian inference to calculate a probability distribution on a range of velocities for each cell requiring to perform calculations for cells which belong to actually static parts of the environment. With current work we will be able to reduce this processing time by limiting to those cells which have been detected as belonging to moving objects only. This will, in turn, result into an improved performance of FCTA algorithm which is based on BOF output.

REFERENCES

[1] C. Laugier, I. Paromtchik, M. Perrollaz, M. Yong, J. Yoder, C. Tay, K. Mekhnacha, and A. Negre, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *Intelligent Transportation Systems Magazine, IEEE*, vol. 3, no. 4, pp. 4 –19, winter 2011.

[2] R. Jain, W. N. Martin, and J. K. Aggarwal, "Segmentation through the detection of changes due to motion," *Computer Graphics and Image Processing*, vol. 11, no. 1, pp. 13–34, September 1979.

[3] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[4] D. Li, "Moving objects detection by block comparison," in *Electronics, Circuits and Systems, 2000. ICECS 2000. The 7th IEEE International Conference on*, vol. 1, 2000, pp. 341–344.

[5] S. Taleghani, S. Aslani, and S. Shiry, *Robust Moving Object Detection from a Moving Video Camera Using Neural Network and Kalman Filter*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 638–648. [Online]. Available: http://dl.acm.org/citation.cfm?id=1575210.1575268

[6] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," vol. 1, 2003.

[7] J. Burlet, T. Vu, and O. Aycard, "Grid-based localization and online mapping with moving objects detection and tracking," INRIA-UJF, Tech. Rep., 2007.

[8] T. Vu, J. Burlet, and O. Aycard, "Grid-based localization and local mapping with moving objects detection and tracking," *International Journal on Information Fusion, Elsevier*, 2009, to appear.

[9] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2, pp. 123–139, 2009.

[10] T.-D. Vu and O. Aycard, "Lased-based detection and tracking moving object using data-driven markov chain monte carlo," in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[11] J. Moras, V. Cherfaoui, and P. Bonnifait, "Moving objects detection by conflict analysis in evidential grids," in *2011 Intelligent Vehicles*

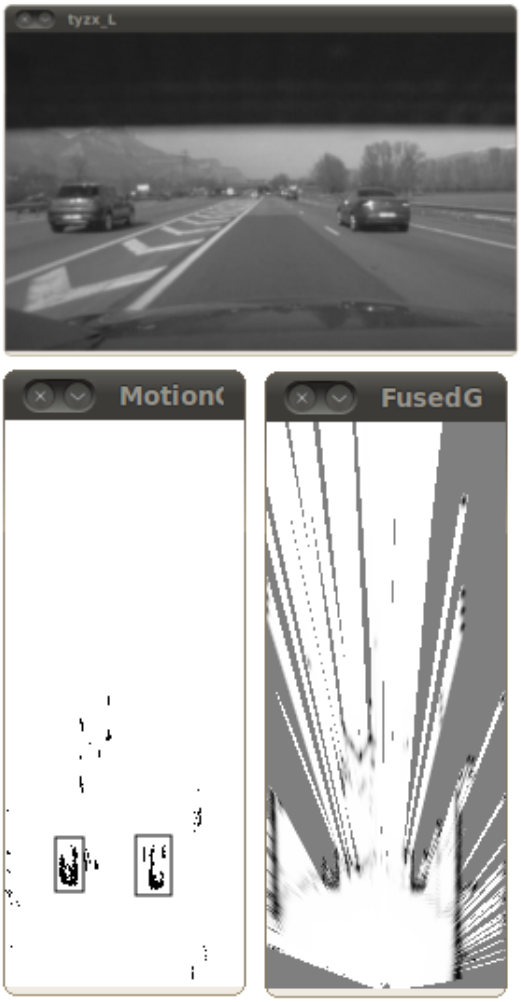


Fig. 8. Motion detection results of two cars on a highway.

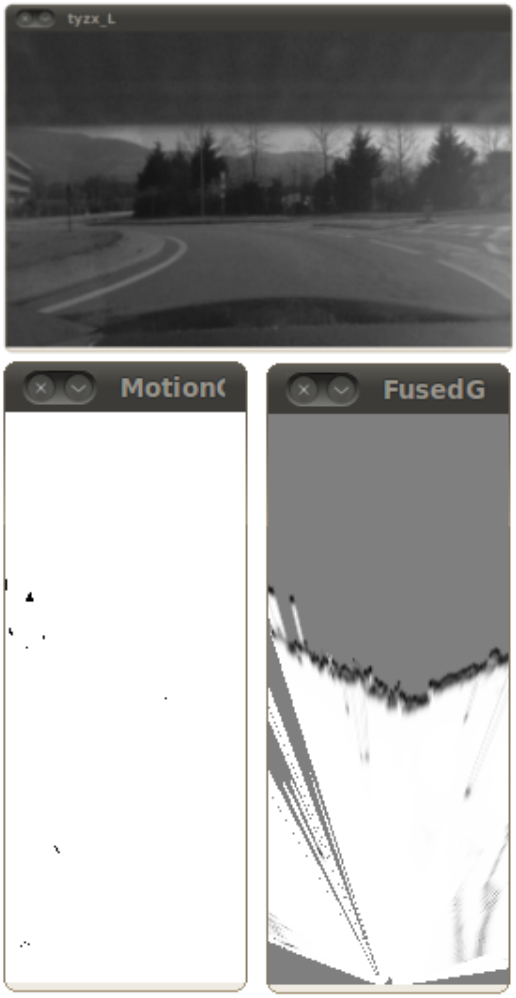


Fig. 9. Motion detection results with no moving object in the view.

Symposium, Baden-Baden, Allemagne, June 5-9 2011, pp. 1120–1125. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00615304/en/

[12] ——, "Credibilist occupancy grids for vehicle perception in dynamic environments," in *2011 IEEE International Conference on Robotics and Automation, Shanghai International Conference Center*, Shanghai, Chine, May 9-13 2011, pp. 84–89. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00615303/en/

[13] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Proceedings of the Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*. New York, NY: Elsevier Science, 1990, pp. 136–146.

[14] C. Tay, K. Mekhnacha, C. Chen, M. Yguel, and C. Laugier, "An efficient formulation of the bayesian occupation filter for target tracking in dynamic environments," 2007, (Accepted) To be published. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/TMCYL07

[15] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian Occupancy Filtering for Multitarget Tracking: an Automotive Application," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, Jan. 2006, voir basilic : http://emotion.inrialpes.fr/bibemotion/2006/CPLFB06/.

[16] K. Mekhnacha, Y. Mao, D. Raulo, and C. Laugier, "Bayesian occupancy filter based "Fast Clustering-Tracking" algorithm," in *IROS 2008*, Nice, France, 2008.

[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*, ser. Intelligent robotics and autonomous agents. The MIT Press, Aug. 2005. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262201623

[18] J. D. Adarve, M. Perrollaz, A. Makris, and C. Laugier, "Computing Occupancy Grids from Multiple Sensors using Linear Opinion Pools," in *IEEE International Conference on Robotics and Automation*, St Paul, Minnesota, États-Unis, May 2012.

[19] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. pp. 118–121, 1974. [Online]. Available: http://www.jstor.org/stable/2285509

[20] C. Tay, K. Mekhnacha, C. Chen, M. Yguel, and C. Laugier, "An Efficient Formulation of the Bayesian Occupation Filter for Target Tracking in Dynamic Environments," *International Journal Of Autonomous Vehicles*, 2007, voir basilic : http://emotion.inrialpes.fr/bibemotion/2007/TMCYL07/.