International workshop on

Safe navigation in open and dynamic environments Application to autonomous vehicles

Full Day Workshop
May 12th 2009, Kobe, Japan

http://wwwlasmea.univ-bpclermont.fr/Control/workshopICRA09/SafeNavigation.htm

Organizers

Christian Laugier (INRIA, France)
Philippe Martinet (IFMA and LASMEA, France),
Urbano Nunes (ISR, Portugal)

Contact

Professor Philippe Martinet
IFMA, Campus des Cezeaux, 63175 Aubiere, France
LASMEA-CNRS Laboratory, Blaise Pascal University
Campus des Cezeaux, 63177 Aubiere, Cedex, France
Phone: +33 473 407 653, Sec : +33 473 407 261, Fax : +33 473 407 262
Email: martinet@lasmea.univ-bpclermont.fr, Home page: http://isrc.skku.edu/~martinet

Forword

The purpose of this workshop is to discuss topics related to the challenging problems of autonomous navigation 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. Such new technologies can also be efficiently applied to other application field such as unmanned vehicles, intelligent wheelchair, service robots, or more generally to human assistance. Technical contributions related to this area, such as autonomous outdoor vehicles, achievements, challenges and open questions will be presented and discussed. Five technical areas, with a focus to their instantiation to dynamic environments, will particularly be addressed: Vision-Based Perception, Multi-sensors Perception & Localisation, SLAM & 3D Reconstruction, Path Planning & Navigation Systems, and Motion Planning.

Previously, two workshops were organized in the same field. The 1st edition of this workshop was held in Roma during ICRA'07 (around 60 attendees), and the second in Nice during IROS'08 (more than 90 registered people). A special issue of IEEE Transactions on ITS will be published at the beginning of 2009 following these two workshops mainly focused on Car and ITS applications. This workshop will be more focused on vision based perception, robust sensor-based navigation, and human robot interaction.

This workshop is composed with 6 invited talks and 12 selected papers. Four sessions has been organized:

1. Session 1: Vision based perception & Visual SLAM
2. Session II: Multi-sensor perception & navigation
3. Session III: SLAM, Localization, Reconstruction
4. Session IV: Motion planning

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

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

Christian Laugier, Philippe Martinet and Urbano Nunes

Session I

Vision based perception & Visual SLAM

- **Title: Comparing appearance-based controllers for nonholonomic navigation from a visual memory** *(invited paper)*
  Authors: Andrea Cherubini, Manuel Colafrancesco, Giuseppe Oriolo, Luigi Freda and François Chaumette

- **Title: A generic framework for topological navigation of urban vehicle**
  Authors: Jonathan Courbon, Youcef Mezouar, Laurent Eck, Philippe Martinet

- **Title: Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments**
  Authors: Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci, Domenico G. Sorrenti

- **Title: Optimal Metric SLAM for Autonomous Navigation Assistance**
  Authors: P.F. Alcantarilla, I. Parra, L.M. Bergasa

ICRA2009

2009 IEEE International Conference
on Robotics and Automation
Kobe, Japan, May 12-17, 2009

Session I

Vision based perception & Visual SLAM

- **Title: Comparing appearance-based controllers for nonholonomic navigation from a visual memory** *(invited paper)*
  Authors: Andrea Cherubini, Manuel Colafrancesco, Giuseppe Oriolo, Luigi Freda and François Chaumette

Comparing appearance-based controllers for nonholonomic navigation from a visual memory

Andrea Cherubini, Manuel Colafrancesco, Giuseppe Oriolo, Luigi Freda and François Chaumette

Abstract— **In recent research, autonomous vehicle navigation has been often done by processing visual information. This approach is useful in urban environments, where tall buildings can disturb satellite receiving and GPS localization, while offering numerous and useful visual features. Our vehicle uses a monocular camera, and the path is represented as a series of reference images. Since the robot is equipped with only one camera, it is difficult to guarantee vehicle pose accuracy during navigation. The main contribution of this article is the evaluation and comparison (both in the image and in the 3D pose state space) of six appearance-based controllers (one pose-based controller, and five image-based) for replaying the reference path. Experimental results, in a simulated environment, as well as on a real robot, are presented. The experiments show that the two image jacobian controllers, that exploit the epipolar geometry to estimate feature depth, outperform the four other controllers, both in the pose and in the image space. We also show that image jacobian controllers, that use uniform feature depths, prove to be effective alternatives, whenever sensor calibration or depth estimation are inaccurate.**

I. INTRODUCTION

In recent research, mobile robot navigation has been often done by processing visual information [1]. This approach can be useful for navigation in urban environments, where tall buildings can disturb satellite receiving and GPS localization, while offering numerous and useful visual features. The most widespread approaches to visual navigation are the *model-based*, and the *appearance-based* approaches, which we shall briefly recall. Model-based approaches rely on the knowledge of a 3D model of the navigation space. The model utilizes perceived features (e.g., lines, planes, or points), and a learning step can be used for estimating it. Conversely, the appearance-based approach does not require a 3D model of the environment, and works directly in the sensor space. The environment is described by a topological graph, where each node corresponds to the description of a position, and a link between two nodes defines the possibility for the robot to move autonomously between the two positions.

In this work, we focus on appearance-based navigation, with a single vision sensor. The environment descriptors correspond to images stored in an *image database*. A similarity score between the view acquired by the camera and the database images, is used as input for the controller that leads the robot to its final destination (which corresponds to a *goal image* in the database). Various strategies can be used to control the robot during navigation. An effective method is visual servoing [2], which was originally developed for manipulator arms, but has also been used for controlling nonholonomic robots (see, for instance, [3]).

The main contribution of this paper is the comparison between six controllers for nonholonomic appearance-based navigation using monocular vision. In particular we investigate the performance of this controllers both in the image, and in the 3D pose state spaces. The paper is organized as follows. In Sect. II, a survey of related works is carried out. In Sect. III, the problem of appearance-based nonholonomic navigation from a visual memory is defined. Although the scope of this paper is the discussion of the control strategies, in Sect. IV, we outline the image processing and the 3D reconstruction algorithms used in our navigation framework. In Sect. V, we present and illustrate the six controllers. The simulated and experimental results are presented in Sect. VI.

A. Cherubini, M. Colafrancesco, G. Oriolo and L. Freda are with the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Ariosto 25, 00185 Roma, Italy {cherubini, oriolo, freda}@dis.uniroma1.it

F. Chaumette is with INRIA-IRISA, Campus de Beaulieu 35042, Rennes, France Francois.Chaumette@irisa.fr

II. RELATED WORK

Recent works in the field of appearance-based autonomous vehicle navigation are surveyed hereby. Most of these works [3 – 13] present a framework with these characteristics:

- a wheeled robot with an on-board camera is considered;
- during a preliminary phase, the *teaching phase*, the robot motion is controlled by a human operator, and a set of images is acquired and stored in a database;
- an *image path* to track is then described by an ordered set of *reference images*, extracted from the database;
- during the *replaying phase*, the robot (starting 'near' the teaching phase initial position) is required to repeat the same path;
- the replaying phase relies on a matching procedure (usually based on correlation) that compares the currently observed image with the reference images;
- although the control strategy enabling the robot to track the learned path varies from one work to the other, it relies, in all cases, on the comparison between the current and reference images.

The methods presented hereby can be subdivided in two main areas. In some works, a three dimensional reconstruction of the workspace is used. The other navigation frameworks, instead, rely uniquely on image information.

We firstly survey the works where 3D reconstruction is utilized. In 1996, Ohno and others [4] propose to use the image database to reconstruct the robot pose in the workspace (i.e., position and orientation) which is utilized for control. In [5], a three dimensional representation of the

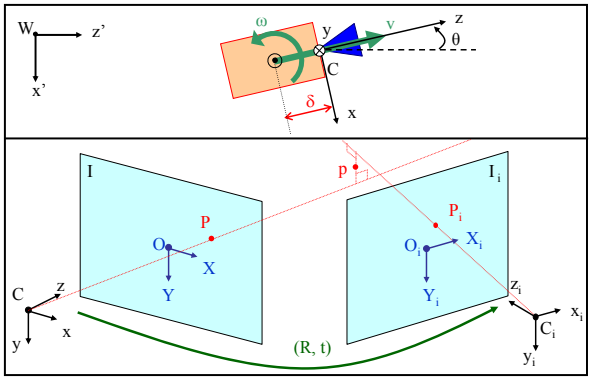


Fig. 1. Relevant variables utilized in this work. Top: mobile robot (orange), equipped with fixed pinhole camera (blue), and applied control variables (v, ω). Bottom: two different views (distinct camera placements) of the same 3-D point p, i.e., in the current (left) and reference (right) images.

taught path is built from the image sequence, and a classic path following controller is used for navigation. Similarly, in [6], pairs of neighboring reference images are associated to a straight line in the 3D workspace, that the robot must track. The epipolar geometry and a planar floor constraint are used to compute the robot heading used for control in [7]. Similarly, in [8], 3D reconstruction is used to improve an omnidirectional vision-based navigation framework.

In general, 3D reconstruction is unnecessary, since moving from one reference image to the next, can also be done by relying uniquely on visual information, as shown in many papers. For instance, in [3], the vehicle velocity commands and the camera pan angle are determined using an image-based visual servoing scheme. In [9], a particular motion (e.g., 'go forward', 'turn left') is associated to each image, in order to move from the current to the next image in the database. In [10], a proportional control on the position of the feature centroid in current and reference images drives the robot steering angle, while the translational velocity is set to a constant value. The controller presented in [11] exploits angular information regarding the features matched in panoramic images. Energy normalized cross correlation is used to control the robot heading in [12]. In [13], a specific image jacobian, relating the change of some image features with the changes in motion in the plane, is used for control.

In summary, a large variety of control schemes has been applied for achieving nonholonomic navigation from a visual memory. However, a comparison between the various approaches has never been carried out. Moreover, in most of the cited articles, the focus has been the qualitative evaluation of the proposed navigation framework in real, complex, environments, without a quantitative assessment of the controller performance. In this paper, we shall compare the performance of six approaches to nonholonomic navigation from a visual memory. The controllers will be assessed using various metrics, both in simulations, and in real experiments. In particular, we will compare the controller accuracy both in the image and in the pose state space, since both are fundamental for precise unmanned navigation.

III. PROBLEM DEFINITION

A. System characteristics

In this work, we focus on a nonholonomic mobile robot of unicycle type, equipped with a fixed pinhole camera. The workspace where the robot moves is planar: $\mathcal{W} = I\!R^2$. With reference to Fig. 1, let us define the reference frames: world frame $\mathcal{F}_\mathcal{W}\,(W, x', z')$, and image frame $\mathcal{F}_\mathcal{I}(O, X, Y)$ (point O is the image plane center). The *robot configuration* is: $q = [x'\ z'\ \theta]^T$, where $[x'\ z']^T$ is the *Cartesian position* of the robot center in $\mathcal{F}_\mathcal{W}$, and $\theta \in\ ]-\pi, +\pi]$ is the *robot heading* (positive counterclockwise) with respect to the world frame z' axis. We choose $u = [v\omega]^T$ as the pair of control variables for our system; these represent respectively the linear and angular velocities (positive counterclockwise) of the robot. The state equation of the robot is:

$$\dot{q} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u$$

We also define the camera frame $\mathcal{F}_\mathcal{C}(C, x, y, z)$, shown in Fig. 1 ($C$ is the optical center). The distance between the y axis and the robot rotation axis is denoted by δ. A pinhole camera model is considered; radial distortion is neglected. Hence, the camera intrinsic parameters are the principal point coordinates and the focal lengths in horizontal and vertical pixel size: f_X, and f_Y. In the following, we consider that the camera parameters have been determined through a preliminary calibration phase, although we shall partially relax this assumption later in the paper. Image processing is based on the grey-level intensity of the image, called $I(P)$ for pixel $P = (X, Y)$.

As outlined in Sect. II, our navigation framework relies on a teaching and on a replaying phases. These phases will be described in the rest of this section.

B. Teaching phase

During the teaching phase, an operator guides the robot stepwise along a continuous path. Each of the N *teaching steps* starts at time t_{i-1} and ends at $t_i > t_{i-1}$ $(i = 1, \ldots, N)$. At each step i, the control input u is assigned arbitrarily by the operator. In this work, we assume that throughout teaching, the robot moves forward, i.e., $v > 0$. At the end of each teaching step, the robot acquires a reference image, that we call I_i, and stores it in a database. Visual features are detected in each I_i. We call $\mathcal{F}_{\mathcal{C}i}(C_i, x_i, y_i, z_i)$ and $\mathcal{F}_{\mathcal{I}i}(O_i, X_i, Y_i)$ (see Fig. 1) the N camera and corresponding N image frames associated to the reference configurations q_i reached at the end of each teaching step.

C. Replaying phase

At the beginning of the replaying phase, the robot is placed at the starting position of the teaching phase. During the replaying phase, the robot must autonomously track the path executed during the teaching phase. The task of replaying the taught path is divided into N subtasks, each consisting of zeroing the visual error between the currently acquired image (called I) and the next reference image $(I_1, I_2, \ldots, I_N)$ in

the database. In practice, as soon as the visual error between I and goal image I_i is 'small enough', the subtask becomes that of reaching image I_{i+1}. Both the visual error and the switching condition will be detailed in Sect. V. Throughout replaying, the linear velocity is fixed to a constant value $\bar{v} > 0$, while the angular velocity ω is derived with a feedback law dependent on the visual features. In all six feedback controllers that we have tested, at each iteration of subtask i, ω is based on the feature points matched between the current image I and the reference image I_i.

IV. VISION ISSUES

A. Image processing

During both teaching and replaying, the images acquired by the robot camera must be processed in order to detect feature points. Besides, during the replaying phase, correspondences between feature points in images I and I_i are required to generate the set of matched points which is used to control the robot. In both teaching and replaying phases, we detect feature points with the well known Harris corner detector [14]. Every iteration of the replaying phase relies on image matching between Harris corners in the current image I and in the nearest next reference image in the database I_i. For each feature point P in image I, we use a correlation technique to select the most similar corresponding point P_i in image I_i. For each pair of images (I, I_i), the algorithm returns the n pairs of *matched points* $(P, P_i)_j$, $j = 1, \ldots, n$.

B. Deriving 3D information

In one the control schemes used in this work (i.e., the *robot heading controller*), it is necessary to estimate the camera pose variation (rotation $\mathbf{R}$ and translation $\mathbf{t}$, see Fig. 1) between the current view I and the next reference view I_i during replay. Moreover, in two of the five image jacobian controllers used, the z coordinates in $\mathcal{F}_\mathcal{C}$ (i.e., the *depths*) of the retroperspective projection p of feature points must be estimated. The depths can also be derived from the camera pose variation. The problem of estimating the camera pose variation ($\mathbf{R}$, $\mathbf{t}$) is a typical *structure from motion* problem.

In some works (see, for instance, [5]), the camera pose is estimated by using bundle adjustment methods, which result in long computation processing, unsuitable for on-line use. Here, we have decided to perform on-line 3D reconstruction, by using only the pair of images (I, I_i), instead of I with the whole database. This choice inevitably implies lower computational time to the detriment of the 3D reconstruction accuracy. The technique that we used for camera pose estimation is epipolar geometry (see [15], for further details). Using an estimate of the distance from q to q_i for $\|\mathbf{t}\|$, four alternative solutions $(\mathbf{R}, \mathbf{t})$ can be derived. For each of the four possible pose variations, we use the technique described in [16] to derive the feature point 3D position p, as the midpoint on the perpendicular to the projecting rays in the two camera frames (see Fig. 1). Finally, we select the pose variation $(\mathbf{R}, \mathbf{t})$ with the greatest number of positive depths in both camera frames $\mathcal{F}_\mathcal{C}$ and $\mathcal{F}_{\mathcal{C}i}$, since feature points must lie in front of both image planes.

V. CONTROL SCHEMES

In this section, we describe the characteristics of the six controllers on ω that we have tested in the replaying phase (v is fixed to constant value $\bar{v}$, see Sect. III). In all cases, we consider that subtask i (i.e., reaching image I_i) is achieved, and we consequently switch to reaching image I_{i+1}, as soon as the average feature error:

$$\epsilon_i = \frac{\sum_{j=1}^{n} \|P_j - P_{i,j}\|}{n}$$

is below a threshold τ_ϵ, and starts to rise.

The first feedback law that we will describe, is *pose-based*: the feedback law is expressed in the robot workspace, by using the 3D data derived from image matching as described in Sect. IV-B. The 5 other feedback laws, instead, are *image-based*: both the control task, and the control law are expressed in the image space, by using the well known image jacobian paradigm. In practice, an error signal measured directly in the image is mapped to actuator commands. Two of the 5 image jacobian controllers require camera pose estimation to derive the depth of feature points. For the 3 others, some approximations on the feature depths are used, as will be shown below.

We hereby recall the image jacobian paradigm which is used by the five image-based controllers. The image jacobian is a well known tool in image-based visual servo control [2], which is used to drive a vector of k visual features s to a desired value s^*. It has been previously applied for solving the problem of nonholonomic appearance-based navigation from a visual memory (see, e.g., [3] and [13]). Let us define:

$$u_c = [v_{c,x}\ v_{c,y}\ v_{c,z}\ \omega_{c,x}\ \omega_{c,y}\ \omega_{c,z}]^T$$

the camera velocity expressed in $\mathcal{F}_\mathcal{C}$. The matrix $\mathbf{L}_\mathtt{S}$ relates the velocity of feature s to u_c:

$$\dot{s} = \mathbf{L}_\mathtt{S} u_c \tag{1}$$

For the robot model that we are considering, the camera velocity u_c can be expressed in function of $u = [v\ \omega]^T$ by using the homogeneous transformation:

$$u_c = {}^\mathtt{C}\mathbf{T}_\mathtt{R} u \tag{2}$$

with:

$${}^\mathtt{C}\mathbf{T}_\mathtt{R} = \begin{bmatrix} 0 & -\delta \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}$$

In the following, we will call T_v and T_ω the first and second columns of ${}^\mathtt{C}\mathbf{T}_\mathtt{R}$. Injecting (2) in (1), we obtain:

$$\dot{s} = \mathbf{L}_{\mathtt{S},v} v + \mathbf{L}_{\mathtt{S},\omega} \omega$$

where $\mathbf{L}_{\mathrm{S},v} = \mathbf{L}_{\mathrm{S}} T_v$, and $\mathbf{L}_{\mathrm{S},\omega} = \mathbf{L}_{\mathrm{S}} T_\omega$ are $k \times 1$ column vectors. In order to drive s to the desired value s^*, we set $v = \bar{v}$ and we select as control law on ω:

$$\omega = -{\mathbf{L}_{\mathrm{S},\omega}}^{+} \left(\lambda e + \mathbf{L}_{\mathrm{S},v} \bar{v}\right) \tag{3}$$

where λ is a given positive gain, e is the error $s - s^*$, and ${\mathbf{L}_{\mathrm{S},\omega}}^{+} \in I\!R^{1\times k}$ is the Moore-Penrose matrix pseudoinverse of $\mathbf{L}_{\mathrm{S},\omega}$, i.e., ${\mathbf{L}_{\mathrm{S},\omega}}^{+} = \left({\mathbf{L}_{\mathrm{S},\omega}}^{T} \mathbf{L}_{\mathrm{S},\omega}\right)^{-1} {\mathbf{L}_{\mathrm{S},\omega}}^{T}$.

A. Robot heading controller

The first controller that we tested in this work, the *robot heading controller* (called RH), is based on the 3D information, derived as described in Sect. IV-B. Since the y-axis is parallel to the robot rotation axis, from the matrix $\mathbf{R}$ defining the rotation between the two camera frames, it is trivial to derive the relative heading variation between the two robot configurations $\Delta\theta = \theta - \theta_i$. Then, we apply the control law:

$$\omega = -\lambda \Delta\theta$$

with λ a given positive gain. A similar controller has been used in [7]. In contrast with that work, however, we do not use the planar constraint to derive the 3D pose variation, and we use $\mathbf{R}$, instead of $\mathbf{t}$ (which is usually more affected by noise), to derive the heading value.

B. Image jacobian points controller

In the *image jacobian points controller* (IJP), the visual features used for achieving subtask i are the current image I coordinates of the n matched points:

$$s = \left[X_1, Y_1, \ldots, Y_n\right]^T \in I\!R^{2n}$$

Each subtask i will consists of zeroing error:

$$e = \left[X_1 - X_{i,1}, Y_1 - Y_{i,1}, \ldots, Y_n - Y_{i,n}\right]^T \in I\!R^{2n}$$

For a normalized perspective camera, the expression of $\mathbf{L}_{\mathrm{P}}$ for a single image point $P\,(X, Y)$ seen in I is:

$$\mathbf{L}_{\mathrm{P}} = \begin{bmatrix} -\frac{1}{z} & 0 & \frac{X}{z} & XY & -1-X^2 & Y \\ 0 & -\frac{1}{z} & \frac{Y}{z} & 1+Y^2 & -XY & -X \end{bmatrix}$$

where z is derived with the method described in Sect. IV-B. By applying the transformation ${}^{\mathrm{C}}\mathbf{T}_{\mathrm{R}}$, we obtain:

$$\mathbf{L}_{\mathrm{P},v} = \begin{bmatrix} \frac{X}{z} \\ \frac{Y}{z} \end{bmatrix} \qquad \mathbf{L}_{\mathrm{P},\omega} = \begin{bmatrix} \frac{\delta}{z} + 1 + X^2 \\ XY \end{bmatrix}$$

If we consider all n matched points between I and I_i, by merely stacking n times vectors $\mathbf{L}_{\mathrm{P},v}$ and $\mathbf{L}_{\mathrm{P},\omega}$, we obtain the two $2n \times 1$ column vectors $\mathbf{L}_{\mathrm{S},v}$ and $\mathbf{L}_{\mathrm{S},\omega}$ to be used in (3)[1].

[1] ${\mathbf{L}_{\mathrm{S},\omega}}^{+}$ is always defined, since:

$${\mathbf{L}_{\mathrm{S},\omega}}^{T} \mathbf{L}_{\mathrm{S},\omega} = \sum_{j=1}^{n} \left[\left(\frac{\delta}{z_j} + 1 + X_j^2 \right)^2 + (X_j Y_j)^2 \right] > 0$$

because $\frac{\delta}{z} + 1 + X^2 > 0$ for all P.

C. Image jacobian points controller with uniform depths

The *image jacobian points controller with uniform depths* (IJPU) is based on an approximation of the model used by the IJP controller. The only difference between the IJPU controller and the IJP controller, is that the depths of all points P_j (which are required for calculating $\mathbf{L}_{\mathrm{S},v}$ and $\mathbf{L}_{\mathrm{S},\omega}$) are assumed identical and set to a fixed value:

$$z_j = \bar{z} \qquad\qquad \forall j = 1, \ldots, n$$

Although this approximation requires $\bar{z}$ to be tuned by the user, depending on the workspace characteristics, and although it can lead to imprecision in the case of sparse 3D points, setting $z_j = \bar{z}$ avoids the need for 3D reconstruction, and consequently spares computational resources. In practice, the IJPU uses an approximation of the interaction matrix similar to the ones commonly used in the visual servoing literature, when pose estimation should be avoided. In fact, it has been shown in many works that a coarse approximation of the image jacobian, without depth estimation, is often sufficient to achieve visual servoing tasks [2], and uniform depths have been successfully used in [17].

D. Image jacobian centroid controller

In the *Image jacobian centroid controller* (IJC), the visual features used for achieving subtask i are the current image I coordinates of the centroid of the n matched points:

$$s = \left[X_G, Y_G\right]^T = \frac{1}{n} \sum_{j=1}^{n} \left[X_j, Y_j\right]^T \in I\!R^2$$

Each subtask i will consists of zeroing error:

$$e = \left[X_G - X_{i,G}, Y_G - Y_{i,G}\right]^T \in I\!R^2$$

For a normalized perspective camera, the expression of $\mathbf{L}_{\mathrm{S}}$ related to the centroid of a discrete set of n image points has been derived in [18] by using image moments:

$$\mathbf{L}_{\mathrm{S}} = \frac{1}{n} \sum_{j=1}^{n} \begin{bmatrix} -\frac{1}{z_j} & 0 & \frac{X_j}{z_j} & X_j Y_j & -1-X_j^2 & Y_j \\ 0 & -\frac{1}{z_j} & \frac{Y_j}{z_j} & 1+Y_j^2 & -X_j Y_j & -\sum_{j=1}^{n} X_j \end{bmatrix}$$

where the z_js are derived with the method described in Sect. IV-B. By applying the transformation ${}^{\mathrm{C}}\mathbf{T}_{\mathrm{R}}$, we obtain:

$$\mathbf{L}_{\mathrm{S},v} = \frac{1}{n} \sum_{j=1}^{n} \begin{bmatrix} \frac{X_j}{z_j} \\ \frac{Y_j}{z_j} \end{bmatrix} \qquad \mathbf{L}_{\mathrm{S},\omega} = \frac{1}{n} \sum_{j=1}^{n} \begin{bmatrix} \frac{\delta}{z_j} + 1 + X_j^2 \\ X_j Y_j \end{bmatrix} \tag{4}$$

These two vectors are used in the control law (3)[2].

[2] ${\mathbf{L}_{\mathrm{S},\omega}}^{+}$ is always defined, since:

$${\mathbf{L}_{\mathrm{S},\omega}}^{T} \mathbf{L}_{\mathrm{S},\omega} = \frac{1}{n^2} \left[\left(\sum_{j=1}^{n} \frac{\delta}{z_j} + 1 + X_j^2 \right)^2 + \left(\sum_{j=1}^{n} X_j Y_j \right)^2 \right] > 0$$

because $\sum_{j=1}^{n} \frac{\delta}{z_j} + 1 + X_j^2 > 0$.

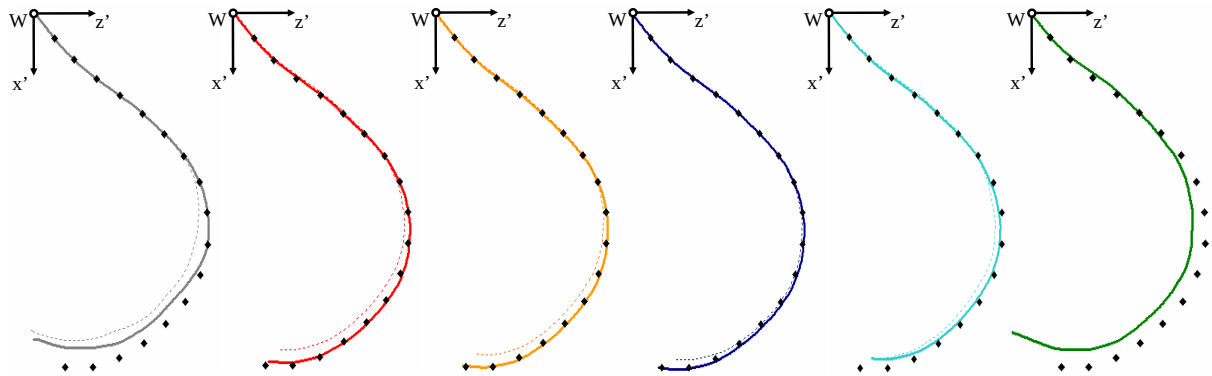


Fig. 2. Robot path in $\mathcal{F}_{\mathcal{W}}$ during Webots navigation, with $N = 17$ reference configurations q_i (black dots), using controllers: RH, IJP, IJPU, IJC, IJCU, AIJCA (left to right), in the experiments with correct (full curves) and coarse (dashed curves) camera calibration.

E. Image jacobian centroid controller with uniform depths

The *image jacobian centroid controller with uniform depths* (IJCU) is based on an approximation of the model used by the IJC controller. The approximation is identical to the one used in IJPU to avoid 3D reconstruction:

$$z_j = \bar{z} \qquad \forall j = 1, \ldots, n$$

with $\bar{z}$ to be tuned according to the workspace characteristics.

F. Approximated image jacobian centroid abscissa controller

From a control viewpoint, since in (3) we control only one degree of freedom (ω), one feature is sufficient for control. In the *approximated image jacobian centroid abscissa controller* (AIJCA), the only visual feature that we use is the abscissa of the centroid of the n matched points in I:

$$s = X_G \in I\!R$$

This choice is reasonable, since the camera optical axis is orthogonal to the robot rotation axis. Each subtask i will then consist of zeroing error:

$$e = X_G - X_{i,G} \in I\!R$$

The relationship between $\dot{s}$ and u_c is here characterized only by the first lines of $\mathbf{L}_{\mathrm{S},v}$ and $\mathbf{L}_{\mathrm{S},\omega}$ in (4). By neglecting δ with respect to the point depths, and assuming that the centroid stays 'near' the image plane center, we assume that:

$$\frac{1}{n}\sum_{j=1}^{n}\frac{X_j}{z_j} \ll 1 \qquad \frac{1}{n}\sum_{j=1}^{n}\frac{\delta}{z_j} + X_j^2 \ll 1$$

which leads to $\mathbf{L}_{\mathrm{S},v} = 0$ and $\mathbf{L}_{\mathrm{S},\omega} = 1$. Replacing in (3) leads to the same control law used in [10]:

$$\omega = -\lambda e$$

where λ is a given positive gain. With this method, no metrical knowledge of the 3-D scene is needed, since the controller relies uniquely on the image features. However, we will show that not taking into account 3D information, is a limitation for this control strategy.

VI. SIMULATIONS AND EXPERIMENTS

The simulations and experiments have been carried out on a MagellanPro robot. This is a differential-drive robot with a caster wheel added for stability. The on-board camera is a 30 Hz Sony EVI-D31, with a resolution of 640×480 pixels. For preliminary simulations, we have made use of Webots[3], where a simulated robot with the same kinematic and sensorial characteristics as MagellanPro has been designed. Video clips of the experiments are available at: `www.dis.uniroma1.it/~labrob/research/VisNav.html`.

In Webots, the six controllers have been compared by replaying a taught path of approximately 4.8 m composed of $N = 17$ reference images; we set $\tau_\epsilon = 4$. Using the Webots GPS sensor, we can derive the 3D paths tracked by the simulated robot in the 6 cases (full curves in Fig. 2). Note that, although with all controllers the robot is able to reach the final goal image I_{17}, path tracking is less accurate with RH and AIJCA than with the 4 other controllers. This result is confirmed by the metrics reported in Table I: both the image error ϵ_i with respect to I_i, and the position error with respect to q_i, averaged over the 17 reference images/configurations, are higher for RH and AIJCA, than for the other controllers. The smaller value of the third metric (average number of matched points n on each image) for RH and AIJCA, is both a cause and an effect of lower accuracy: less points provide less information for control, while, inprecise path tracking worsens feature tracking. Although the performances of the 4 other controllers are comparable, slightly better results are obtained when the depth is estimated using 3D reconstruction (IJP and IJC), than when it is fixed (IJPU and IJCU). The importance of the – mainly longitudinal – position error in RH and AIJCA is due to the fundamental role played by the point depths (which are not used by the latter controllers) in the pose accuracy associated to an image-based task: with RH and AIJCA, the robot stops much after configuration q_{17}. To further investigate the controller performances, we have plotted, in Fig. 3 (left), the typical evolution of ω and ϵ_i during a path step. Here, we focus on the step from I_6 to I_7, although the trends are similar for the other 16 steps. The curves show that with RH, which is merely position-based, the value of ω is strongly conditioned by the 3D

[3] www.cyberbotics.com

TABLE I

CONTROLLERS PERFORMANCE IN WEBOTS - CORRECT CALIBRATION

controller	RH	IJP	IJPU	IJC	IJCU	AIJCA
average ϵ_i w.r.t. I_i (pixels)	2.9	1.9	2.4	2.2	2.6	3.1
average position error (cm)	14	4	5	4	6	21
average n	77	94	92	93	92	73

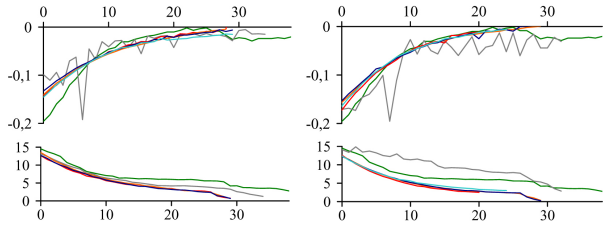


Fig. 3. Evolution of ω (top, in rad/s) and ϵ_7 (bottom, in pixels) at successive iterations while the simulated robot moves from I_6 to I_7 using: RH (grey), IJP (red), IJPU (orange), IJC (blue), IJCU (cyan), and AIJCA (green), with correct (left) and coarse (right) camera calibration.

reconstruction error, and oscillates, leading to later convergence of ϵ_i (hence to the late robot stop). The inaccuracy in $\Delta\theta$ ($\pm 6^\circ$ average estimation error over a total heading variation, throughout the path, of -110°) is due to our choice of estimating on-line the camera pose by using only pairs of images, instead of performing a computationally costly global bundle adjustment. This is consistent with the choice of processing the same sensor input for all controllers, i.e., simply data from the current and from the next reference images. Late convergence of ϵ_i also occurs with AIJCA (green in Fig. 3). Smoother curves are obtained with the other 4 image jacobian controllers, which take into account the feature point image positions, as well as their 3D depths.

To verify the controllers' robustness, the 6 simulations have been repeated with a random calibration error of either $+10\%$ or -10% on each of the camera parameters: f_X, f_Y, δ. For the uniform depth controllers (IJPU and IJCU), we have also included a random calibration error of $+10\%$ or -10% on $\bar{z}$, simulating imprecise tuning of this parameter. For the coarsely calibrated simulations, the replayed paths are represented by the dashed curves in Fig. 2, while the relevant metrics, and the evolution of ω and ϵ_i during the seventh step are shown respectively in Table II, and Fig. 3 (right). For AIJCA, which is independent from the camera parameters, the results are identical to those of the calibrated case. Fig. 2 shows that the robot is able to successfully follow the path in all 6 cases, although path tracking is obviously less precise than in the calibrated camera case. Again, the 4 image jacobian controllers that utilize feature depth, outperform RH (where camera parameters are crucial for control) and AIJCA.

To evaluate the effect of the choice of the parameter $\bar{z}$ used by the two uniform depth controllers, we have repeated the calibrated camera simulations by varying the value of $\bar{z}$ for a fixed gain λ. Since in our workspace the feature points are very sparse (with average depth 1.9 m, and standard deviation 1.5 m), the uniform depth assumption is quite strong. Nevertheless, all the simulations that we have run using $\bar{z} \in [0.8, 500]$ m were successful, and provided good performances: average $\epsilon_i < 3.5$, $n > 80$ and position error < 25 cm. In fact, for $\bar{z} \to \infty$, both IJPU and IJCU rely uniquely on image features, since $\mathbf{L}_{\mathrm{S},v} \to 0$, and $\mathbf{L}_{\mathrm{S},\omega}$

TABLE II

CONTROLLERS PERFORMANCE IN WEBOTS - COARSE CALIBRATION

controller	RH	IJP	IJPU	IJC	IJCU	AIJCA
average ϵ_i w.r.t. I_i (pixels)	3.5	2.7	2.7	2.3	2.5	3.1
average position error (cm)	19	7	7	5	9	21
average n	57	94	96	93	90	73

depends only on the image coordinates of the P_j points; hence, in this case, inappropriate tuning of $\bar{z}$ does not worsen the controllers' performance. On the other hand, for $\bar{z} < 0.8$ m, the simulations fail, due to the large modeling error in the choice of $\bar{z}$, which should be closer to the average value 1.9 m. Therefore, the simulations show that IJPU and IJCU are robust to large $\bar{z}$ modeling errors, and that overestimating $\bar{z}$ is preferable.

To assess the convergence domain, in a fourth series of simulations, the 6 controllers have been tested starting from an initial configuration 'distant' from the teaching phase initial configuration. The distance is evaluated by considering the ratio ρ obtained by dividing the initial image error ϵ_1 (with respect to I_1) in the presence of initial pose error, by the initial ϵ_1 in the ideal case (i.e., when replay starts at the teaching initial configuration). For each controller, we assess the convergence domain by verifying the maximum ρ tolerated. For IJP and IJC, a maximum ρ of 4.1 is tolerated (i.e., these controllers converge from an initial view with ϵ_1 4.1 times larger than the initial teaching view). For IJPU and IJCU, $\rho = 2.6$ is tolerated; for AIJCA and RH, respectively $\rho = 2.1$ and $\rho = 1.9$. Clearly, a complete stability analysis would be required to precisely assess the convergence domain. Nevertheless, these simulation results are useful to confirm the properties of the 6 controllers, and show that IJP and IJC can converge even in the presence of a large initial error.

After the simulation results, we ported the navigation framework on the real MagellanPro for further validation. Since the image jacobian points and centroid controllers have behaved similarly in Webots, we have not tested the centroid controllers IJC and IJCU on the real robot. A taught path of approximately 2.0 m, composed of $N = 4$ reference images, has been replayed using the other 4 controllers, with $\tau_\epsilon = 5$. Since the robustness of the image processing algorithms is not crucial in this work, the environment was lightly structured, by adding artificial visual textures. With RH, the experiment failed after having reached image I_2. The reason is the large position error with respect to the taught path, which causes feature point loss. The replayed paths, are shown, along with the taught path (white) in Fig. 4. Values of the main metrics are reported in Table III, and the evolution of ω and ϵ_i while the robot approaches I_1 are shown in Fig. 5. The experiments confirm the controllers' characteristics seen in Webots. Indeed, both the attempt of accomplishing an image-based task by using merely 3D features (RH), and that of tracking accurately the 3D path by using merely image features (AIJCA) are unfruitful, while the two complete image jacobian controllers provide the best performances both in the image and in the 3D state space. Again, IJP, which utilizes computed depths, outperforms IJPU, which utilizes an approximation of the depths. This result is even

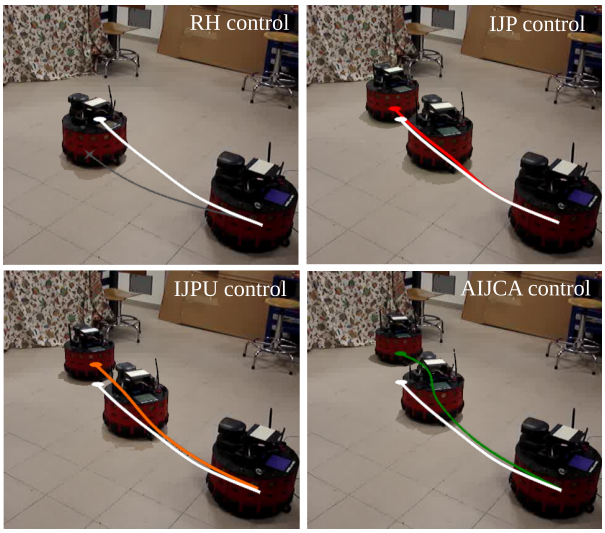


Fig. 4. Replaying a taught path (white) using controllers: RH (grey), IJP (red), IJPU (orange), and AIJCA (green). Robot key positions during navigation are also shown: initial, intermediate and, for the 3 successful controllers, final positions.

more evident in a real environment than in simulations. Fig. 4 also confirms that in all cases, the predominant component of the position error is in the longitudinal direction (i.e., in the z direction), as outlined in the simulations. This is not a surprise, since it is well known that for nonholonomic systems, set-point regulation (which cannot be achieved via smooth time-invariant feedback) is more difficult to achieve than trajectory tracking. Besides, the importance of the longitudinal position error is due to the utilization of scale-dependent Harris points, which are hard to track when the motion in the optical axis direction is important. However, since the object of this study is the control, rather than the sensing technique, this is not crucial: using scale-invariant features will improve navigation, without modifying the controllers' characteristics.

VII. CONCLUSIONS AND FUTURE WORK

We have compared 6 appearance-based controllers for nonholonomic navigation from a visual memory. The simulations and experiments have shown that the 4 complete image jacobian controllers, which combine both image data and feature depth, outperform the 2 controllers which utilize only 3D data, or only image data. Besides, although 3 controllers necessitate 3D reconstruction, for the image jacobian controllers (IJP and IJC), a large 3D reconstruction error (e.g., due to coarse camera calibration) can be allowed without jeopardizing performance. Indeed, in the IJP and IJC experiments, as opposed to the RH experiments, 3D reconstruction performed on-line by using only pairs of subsequent images gave excellent results. Moreover, since 3D reconstruction introduces computational delay at run time, and increases sensitivity to image noise, a valid alternative is to use the uniform depth controllers IJPU and IJCU. We hope that the results of this study can be useful for the researchers working on similar visual navigation frameworks worldwide. Future work will be devoted to taking into account environment modifications between the teaching and replaying phases. We also plan to implement and integrate obstacle avoidance, by considering cases where the robot must deviate from the taught path in order to avoid an obstacle, while maintaining localization accuracy.

TABLE III

COMPARING FOUR CONTROLLERS ON THE REAL ROBOT

controller	RH[a]	IJP	IJPU	AIJCA
average ϵ_i w.r.t. I_i (pixels)	4.2	3.0	3.8	4.5
average position error (cm)[b]	40	30	33	44
average n	42	63	57	32

[a]Since RH failed after I_2, these are averaged over 2 replay steps.
[b]Estimated from the videos of the experiments.

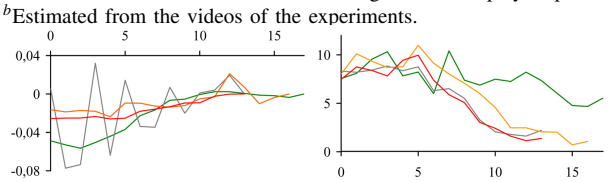


Fig. 5. Evolution of ω (left, in rad/s) and ϵ_1 (right, in pixels) while the robot moves towards I_1 using controllers: RH (grey), IJP (red), IJPU (orange), and AIJCA (green).

REFERENCES

[1] G. N. DeSouza and A. C. Kak, "Vision for Mobile Robot Navigation: a Survey", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, 2002.
[2] F. Chaumette and S. Hutchinson, "Visual servo control tutorial, part I and II", *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, and vol. 14, no. 1, 2007.
[3] Y. Masutani, M. Mikawa, N. Maru and F. Miyazaki, "Visual servoing for non-holonomic mobile robots", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1994.
[4] T. Ohno, A. Ohya and S. Yuta, "Autonomous navigation for mobile robots referring pre-recorded image sequence", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1996.
[5] E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation", *Int. Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.
[6] G. Blanc, Y. Mezouar and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes", *IEEE Int. Conf. on Robotics and Automation*, 2005.
[7] O. Booij, B. Terwijn, Z. Zivkovic, B. Kröse, "Navigation using an appearance based topological map", *IEEE Int. Conf. on Robotics and Automation*, 2007.
[8] T. Goedemé, M. Nuttin, T. Tuytelaars and L. Van Gool, "Omnidirectional vision based topological navigation", *Int. Journal of Computer Vision*, vol. 74, no. 3, pp. 219–236, 2007.
[9] Y. Matsumoto, M. Inaba and H. Inoue, "Visual navigation using view-sequenced route representation", *IEEE Int. Conf. on Robotics and Automation*, 1996.
[10] A. Diosi, A. Remazeilles, S. Šegvić and F. Chaumette, "Outdoor Visual Path Following Experiments", *IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 2007.
[11] A. A. Argyros, K. E. Bekris, S. C. Orphanoudakis and L. E. Kavraki, "Robot homing by exploiting panoramic vision", *Autonomous Robots*, vol. 19, no. 1, pp. 7-25, 2005.
[12] S. S. Jones, C. Andersen and J. L. Crowley, "Appearance based processes for visual navigation", *IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 1997.
[13] D. Burschka and G. Hager, "Vision-based control of mobile robots", *IEEE Int. Conf. on Robotics and Automation*, 2001.
[14] C. Harris and M. Stephens, "A combined corner and edge detector", *4th Alvey Vision Conference*, pp. 147–151, 1988.
[15] Y. Ma, S. Soatto, J. Košecká and S. S. Sastry, "An Invitation to 3-D Vision: from Images to Geometric Models". New York: Springer, 2003.
[16] P. A. Beardsley, A. Zisserman and D. W. Murray, "Sequential Updating of Projective and Affine Structure from Motion", *Int. Journal of Computer Vision*, vol. 23, pp. 235–259, 1997.
[17] A. Remazeilles and F. Chaumette, "Image-based robot navigation from an image memory", *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 345–356, 2007.
[18] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects", *IEEE Trans. on Robotics*, vol. 21, no. 6, pp. 1116–1127, 2005.

Session I

Vision based perception & Visual SLAM

- **Title: A generic framework for topological navigation of urban vehicle**
  Authors: Jonathan Courbon, Youcef Mezouar, Laurent Eck, Philippe Martinet

A generic framework for topological navigation of urban vehicle

Jonathan Courbon*†, Youcef Mezouar†, Laurent Eck*, Philippe Martinet†
*CEA, List
18 route du Panorama, BP6
F- 92265 FONTENAY AUX ROSES - FRANCE
Email: laurent.eck@cea.fr

†LASMEA
24 Avenue des Landais
63177 AUBIERE - FRANCE
Email: firstname.lastname@lasmea.univ-bpclermont.fr

***Abstract*—In this paper, we present a generic framework for urban vehicle navigation using a topological map. This map is built by taking into account the non-holonomic behaviour of the vehicle. After a localization step, a sensory route is extracted to reach a goal. This route is followed using a sensor-based control strategy, based on the vehicle model and computed from the state extracted from the current and the desired sensory images. In that aim, a generic model is proposed for visual sensors. Experiments with an urban electric vehicle navigating in an outdoor environment have been carried out with a fisheye camera using a single camera and natural landmarks. A navigation along a 1700-meter-long trajectory validates our approach.**

***Index Terms*—Urban vehicle navigation, topological map, generic camera model, autonomous navigation, non-holonomic mobile vehicle, real-time application**

I. Introduction

Saturation of vehicles traffic in large cities is a major concern. Improvements can be gained from the development of alternative public transportation systems. In order to meet public expectation, such systems should be very flexible, in order to be suitable answer to many different individual needs, and as nuisance free as possible (with respect to pollution, noise, urban scenery, ...). Individual vehicles, available in a car-sharing concept, meet clearly both requirements. They appear to be very suitable in specific areas where the public demand is properly structured, as in airport terminals, attraction resorts, university campus, or inner-cities pedestrian zones. In order to spread such a transportation system, automatic navigation of those vehicles has to be addressed: passengers could then move from any point to any other point at their convenience in an automatic way, and vehicles could be brought back autonomously to stations for refilling and reuse. Automatic navigation is generally divided in four steps : 1) map building, 2) localisation onto the map, 3) path planning and 4) control to actually achieve the navigation task. Many works deal with the problems of fuzzing steps 1) and 2) on a single stage (Simultaneous Localization And Mapping; SLAM). Unfortunately, even if computers are more and more powerfull, those strategies are restricted to small environments since the computational cost highly increases with the number of features integrated onto the map. An alternative solution, suitable for large scale environment, consists on using a Geographical Information System (GIS) as proposed in [1].

Using visual sensors, appearance-based or "visual memory-based" navigation approaches are emerging. The main idea is to represent the mobile robot environment with a bounded quantity of images gathered in a database (visual memory). For example, [2] proposes to use a sequence of images recorded during a human teleoperated motion, and called View-Sequenced Route Reference. Such a strategy is called "mapless" (refer to [3]). Indeed, any notion of map nor topology of the environment appears, neither to build the reference set of images, nor for the automatic guidance of the mobile robot. Similar approaches have been proposed for urban vehicles in [4]. In practise, a topological organization decreases the computational cost and is more intuitive.

In this paper, we present a generic framework for urban vehicle navigation using a topological map. This topological map directly takes into account the control constraints during its building (refer to Section II). Before the beginning of the motion the localization of the robotic system is performed. Given an image of one of the paths as a target, the vehicle navigation mission is defined as a concatenation of path subsets, called sensory route. A navigation task then consists in autonomously executing this route. The path-following control law adapted to the nonholonomic constraints of the vehicle is first defined (Section III-B). This control guides the vehicle along the reference route without explicitly planning any trajectory. This step requires also a model of the sensor to compute the state needed by the control law. In the case of visual sensors, we propose a generic model valid for a large set of cameras (including perspective, catadioptric, spherical and fisheye cameras). Those elements are presented in Section III-C. Experiments have been carried out with an electrical urban vehicle, navigating in outdoor environment along a 1200-meter-long trajectory. Results are presented in the last Section.

II. Topological map with control constraints

In the sequel, we define an image I as the representation of the environment given by an embedded sensor. This sensor is supposed to be rigidly fixed to the vehicle. The environment is represented by a set of images, topologically organized.

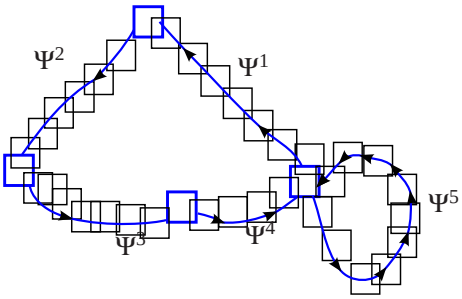


Fig. 1: The memory of the robot, composed of 5 ordered paths Ψ^i. Each square represents an image of the memory.

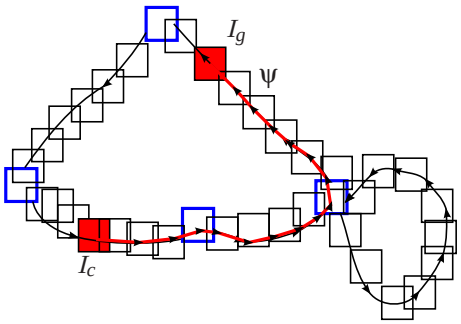


Fig. 2: A sensory route, from the current and goal locations (in the sensory memory) of the robot.

A. Representation of the environment

Let consider a sensory path Ψ^p composed of n key sensory images:

$$\Psi^p = \left\{ I_i^p | i = \{1,2,\ldots,n\} \right\}$$

Such a path is a directed graph composed of images successively acquired. In practise, such a path can represent a street between two crossroads. This representation is justified because, when following the given path it is not necessary to take into account the other elements of the environment. Paths are then linked. The choice of the key images and the path linking is explained in Section II-C. The environment is thus represented by a topological map which is a multigraph of sensory paths (refer to Fig. 1).

B. Paths acquisition

The learning stage relies on the human experience. The user guides the vehicle along one or several paths of its workspace. During this stage, the motions are assumed to be limited to those of a car-like vehicle, which only goes forward. Images are acquired by the embedded sensor and then, a selection process occurs in order to keep only some images called "key images". As noticed in [2], the number of key images of a visual path is directly linked to the human-guided path complexity.

C. Control constraints during map building

For control purpose (refer to Section III), the authorized motions are assumed to be limited to those of a car-like vehicle, which only goes forward. The following Hypothesis 2.1 formalizes these constraints.

Hypothesis 2.1: Given two frames ${}^R\mathcal{F}_i$ and ${}^R\mathcal{F}_{i+1}$, respectively associated to the vehicle when two successive key images I_i and I_{i+1} of a sensory path Ψ were acquired, there exists an admissible path ψ from ${}^R\mathcal{F}_i$ to ${}^R\mathcal{F}_{i+1}$ for a car-like vehicle whose turn radius is bounded, and which only moves forward.

Moreover, because the controller is sensor-based, the robot is controllable from I_i to I_{i+1} only if the hereunder Hypothesis 2.2 is respected.

Hypothesis 2.2: Two successive key sensory images I_i and I_{i+1} contain a set $\mathcal{P}_i$ of matched features, which can be tracked along a path performed between ${}^R\mathcal{F}_i$ and ${}^R\mathcal{F}_{i+1}$ and which are sufficient to compute the full control law.

This Hypothesis 2.2 has three effects. Firstly, it limits the set of possible sensors. In fact, some sensors may not provide enough information to compute the control law. Secondly, for the same reason, the position of the sensor is important as it must provide the needed information. Finally, during the acquisition of a sensory path, this hypothesis constrains the choice of the key images.
In order to connect two sensory paths, the terminal extremity of one of them and the initial extremity of the other one must be constrained as two consecutive key images of a sensory path.

D. Sensory route

A sensory route describes the vehicle's mission in the sensor space. Given two key images of the sensory memory I_c and I_g, corresponding respectively to the current and goal locations of the robot, a sensory route ψ is a set of key images which describes a path from I_c to I_g (refer to Fig.2).
The sensory route describes a set of consecutive states that the sensor has to reach in order that the robot joins the goal configuration from the initial one. The robot motions are controlled along the sensory route using the data provided by the embedded sensor. In that aim, the sensor has to be modelled as well as the vehicle. A control law is then designed and computable by the state given by the sensor's relative information. The next section deals with those issues.

III. Modelling and control

When starting the autonomous navigation task, the output of the localization step provides the closest image I_c to the current initial image I_c^*. A visual route Ψ connecting I_c to the goal image is then extracted from the visual memory. As previously explained, the sensory route is composed of a set of key images. The next step is to automatically follow this route using a sensor-based technique. The principle is presented in Fig. 3.
To design the controller, described in the sequel, the key images of the reference route are considered as consecutive checkpoints to reach in the sensor space. The control problem

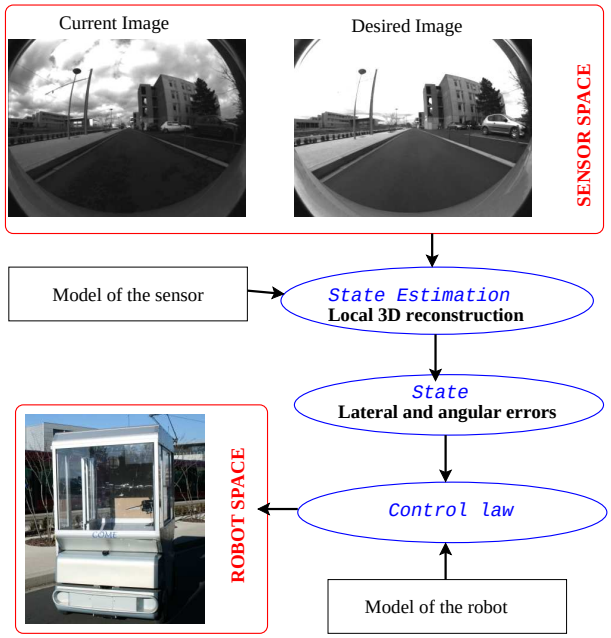


Fig. 3: Route following process with a visual sensor.

is formulated as a path following to guide the nonholonomic vehicle along the sensory route.

A. Model and assumptions

1) Control objective: Let I_i and I_{i+1} be two consecutive key images of a given route to follow and I_c be the current image. Let us note $\mathcal{F}_i = (O_i, \mathbf{X_i}, \mathbf{Y_i}, \mathbf{Z_i})$ and $\mathcal{F}_{i+1} = (O_{i+1}, \mathbf{X_{i+1}}, \mathbf{Y_{i+1}}, \mathbf{Z_{i+1}})$ the frames attached to the vehicle when I_i and I_{i+1} were stored and $\mathcal{F}_c = (O_c, \mathbf{X_c}, \mathbf{Y_c}, \mathbf{Z_c})$ a frame attached to the vehicle in its current location. Figure 4 illustrates this setup. The origin O_c of $\mathcal{F}_c$ is on the center rear axle of a car-like vehicle, which moves on a perfect ground plane. The hand-eye parameters (*i. e.* the rigid transformation between $\mathcal{F}_c$ and the frame attached to the camera) are supposed to be known. According to Hypothesis 2.2, the state of a set of features $\mathcal{P}_i$ is known in the images I_i and I_{i+1}. The state of $\mathcal{P}_i$ is also assumed available in I_c. The task to achieve is to drive the state of $\mathcal{P}_i$ from its current value to its value in I_{i+1}. Let us note Γ a path from $\mathcal{F}_i$ to $\mathcal{F}_{i+1}$. The control strategy consists in guiding I_c to I_{i+1} by regulating asymptotically the axle $\mathbf{Y}_c$ on Γ. The control objective is achieved if $\mathbf{Y}_c$ is regulated to Γ before the origin of $\mathcal{F}_c$ reaches the origin of $\mathcal{F}_{i+1}$.

2) Vehicle Modelling: Our experimental vehicle is devoted to urban transportation, i.e. it moves on asphalt even grounds at rather slow speeds. Therefore, it appears quite natural to rely on a kinematic model, and to assume pure rolling and non slipping at wheel-ground contact. In such cases, the vehicle modelling is commonly achieved for instance relying on the Ackermann's model, also named the bicycle model: the two front wheels located at the mid-distance between actual front wheels and actual rear wheels. As seen previously, our control problem has as objective that the vehicle follows a reference path Γ, we propose to describe here, its configuration with

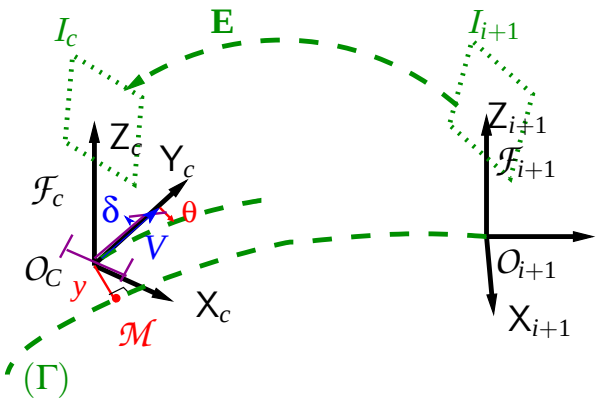


Fig. 4: Images I_i and I_{i+1} are two consecutive key images of the visual route Ψ. I_c is the current image. Γ is the path to follow.

respect to that path, rather than with respect to an absolute frame. To meet this objective, the following notations are introduced (see Figure 4).

- O_C is the center of the vehicle rear axle,
- $\mathcal{M}$ is the point of Γ which is the closest to O_C. This point is assumed to be unique which is realistic when the vehicle remains close from Γ.
- s is the curvilinear coordinate of point $\mathcal{M}$ along Γ and $c(s)$ denotes the curvature of Γ at that point.
- y and θ are respectively the lateral and angular deviation of the vehicle with respect to reference path Γ
- δ is the virtual front wheel steering angle
- V is the linear velocity along the axle $\mathbf{Y_c}$ of $\mathcal{F}_c$
- l is the vehicle wheelbase.

Vehicle configuration can be described without ambiguity by the state vector $(s\ ,\ y\ ,\ \theta)$: the two first variables provide point O_C location and the last one the vehicle heading. Since V is considered as a parameter, the only control variable available to achieve path following is δ. The vehicle kinematic model can then be derived by writing that velocity vectors at point O_C and at center of the front wheel are directed along wheel planes and that the vehicle motion is, at each instant, a rotation around an instantaneous rotation center. Such calculations lead to (refer to [5]):

$$\begin{cases} \dot{s} = V\dfrac{\cos\theta}{1-c(s)y} \\ \dot{y} = V\sin\theta \\ \dot{\theta} = V\left(\dfrac{\tan\delta}{l} - \dfrac{c(s)\cos\theta}{1-c(s)y}\right) \end{cases} \tag{1}$$

Model (1) is clearly singular when $y = \dfrac{1}{c(s)}$ *i.e.* when point O_C is superposed with the path Γ curvature center at abscissa s. However, this configuration is never encountered in practical situations: on one hand, the path curvature is small and on the other, the vehicle is expected to remain close to Γ.

B. Control Design

The control objective is to ensure the convergence of y and θ toward 0 before the origin of $\mathcal{F}_c$ reaches the origin of $\mathcal{F}_{i+1}$. The vehicle model (1) is clearly nonlinear. However, it has been established in [6] that mobile robot models can generally be converted in an exact way into almost linear models, named chained forms. This property offers two very attractive features: on one hand, path following control law can be designed and tuned according to celebrated Linear System Theory, while controlling nevertheless the actual non linear vehicle model. Control law convergence and performances are then guaranteed whatever the vehicle initial configuration is. On the other hand, chained form enables to specify, in a very natural way, control law in term of distance covered by the vehicle, rather than in term of time. Vehicle spacial trajectories can then easily be controlled, whatever the vehicle velocity is [7].

Conversion of the vehicle model (1) into chained form can be achieved thanks to the following state and control transformation:

$$\Phi(\begin{bmatrix} s & y & \theta \end{bmatrix}) \stackrel{\Delta}{=} \begin{bmatrix} s & y & (1-c(s)y)tan(\theta) \end{bmatrix} \quad (2)$$

The expression of the actual control law δ can be obtained by inverting the chained transformation:

$$\begin{aligned} \delta(y,\theta) = \arctan\Big(-l\Big[\tfrac{\cos^3\theta}{(1-c(s)y)^2}\Big(\tfrac{dc(s)}{ds}y\tan\theta \\ -K_d(1-c(s)y)\tan\theta \\ -K_p y + c(s)(1-c(s)y)\tan^2\theta\Big) + \tfrac{c(s)\cos\theta}{1-c(s)y}\Big]\Big) \end{aligned} \quad (3)$$

The gains (K_d, K_p) impose a settling distance and set the desired control performances. Consequently, for a given initial error, the vehicle trajectory will be identical, whatever the value of V is, and even if V is time-varying ($V \neq 0$). Control law performances are therefore velocity independent. In our experiments the path to follow is simply defined as the straight line $\Gamma' = (O_{i+1}, \mathbf{Y}_{i+1})$ (refer to Figure 4). In this case $c(s) = 0$ and the control law (3) can be simplified as follows:

$$\delta(y,\theta) = \arctan\left(-l\left[\cos^3\theta\left(-K_d\tan\theta - K_p y\right)\right]\right) \quad (4)$$

The implementation of control law (4) requires the on-line estimation of the lateral deviation y and the angular deviation θ of $\mathcal{F}_c$ with respect to Γ. In the next Section, we describe how geometrical relationships between two views acquired with a camera under the generic projection model (conventional, catadioptric and fisheye cameras) are exploited to enable a partial Euclidean reconstruction from which (y, θ) are derived.

C. State estimation from a visual sensor

Different sensors are suitable for our application. The method consists on two steps: 1/ sensor modeling, 2/ extraction of the state of the robot in the sensor space. In the sequel, visual cameras are used to extract the state required by the control law but our framework is not limited to those sensors. We consider a camera modeled by the generic projection on the sphere and the image of points features. The unified projection model consists on a projection onto a virtual unitary sphere, followed by a perspective projection onto an image plane [8]. This virtual unitary sphere is centered in the principal effective view point and the image plane is attached to the perspective camera.

Let $\mathcal{F}_c$ and $\mathcal{F}_m$ be the frames attached to the conventional camera and to the unitary sphere respectively. In the sequel, we suppose that $\mathcal{F}_c$ and $\mathcal{F}_m$ are related by a simple translation along the Z-axis ($\mathcal{F}_c$ and $\mathcal{F}_m$ have the same orientationThe origins $\mathcal{C}$ and $\mathcal{M}$ of $\mathcal{F}_c$ and $\mathcal{F}_m$ will be termed optical center and principal projection center respectively. The optical center $\mathcal{C}$ has coordinates $[0\ 0\ -\xi]^T$ with respect to $\mathcal{F}_m$ and the image plane is orthogonal to the Z-axis and it is located at a distance $Z = f_c$ from $\mathcal{C}$.

Let $\mathcal{X}$ be a 3D point with coordinates $\mathsf{X} = [X\ Y\ Z]^T$ with respect to $\mathcal{F}_m$. The point on the normalized image plane is of homogeneous coordinates $\underline{\mathsf{x}} = [\mathsf{x}^T\ 1]^T = f(\mathsf{X})$ (where $\mathsf{x} = [x \quad y]^T$):

$$\underline{\mathsf{x}} = f(\mathsf{X}) = \begin{bmatrix} \dfrac{X}{\varepsilon_s Z + \xi\rho} & \dfrac{Y}{\varepsilon_s Z + \xi\rho} & 1 \end{bmatrix}^\top \quad (5)$$

The parameter ε_s allows to integrate the spherical projection into this model by setting $\varepsilon_s = 0$ and $\xi = 1$. In the general case and in the sequel, this parameter is equal to 1. Note that, setting $\xi = 0$ (and $\varepsilon_s = 1$), the general projection model becomes the well known pinhole model. ξ can be seen as a parameter which allows to control the amount of radial distortions for fisheye lenses.

Finally the point of homogeneous coordinates m in the image plane is obtained after a plane-to-plane collineation $\mathbf{K}$ of the 2D projective point of coordinates $\underline{\mathsf{x}}$:

$$\mathsf{m} = \mathbf{K}\underline{\mathsf{x}} \quad (6)$$

The matrix $\mathbf{K}$ can be written as $\mathbf{K} = \mathbf{K_p}\mathbf{M}$ where the matrix $\mathbf{K_p}$ contains the perspective camera intrinsic parameters, and the diagonal matrix $\mathbf{M}$ links the frame attached to the unitary sphere to the camera frame $\mathcal{F}_m$. For a central catadioptric camera, this matrix depends on the shape of the mirror.

Let $\mathcal{X}$ be a 3D point with coordinates $\mathsf{X_c} = [X_c\ Y_c\ Z_c\ ]^T$ in the current frame $\mathcal{F}_c$ and $\mathsf{X}^* = [X_{i+1}\ Y_{i+1}\ Z_{i+1}\ ]^T$ in the frame $\mathcal{F}_{i+1}$. Let $\mathsf{X_m}$ and X^*_m be the coordinates of those points, projected onto the unit sphere (refer to Fig. 5). Let $\mathbf{R}$ (respectively t) represent the rotational matrix (resp. the translational vector) between the current and the desired frames. Similarly to the case of pinhole model, the epipolar geometry leads to:

$$\mathsf{X_m}^T\mathbf{E}\mathsf{X_m}^{*T} = 0 \quad (7)$$

where $\mathbf{E} = \mathbf{R}[\mathsf{t}]_\times$ is the essential matrix [9]. The essential matrix $\mathbf{E}$ between two images is estimated using five couples of matched points as proposed in [10] if the camera calibration (matrix $\mathbf{K}$) is known. Outliers are rejected using a random sample consensus (RANSAC) algorithm. From the essential matrix, the camera motion parameters (that is the rotation $\mathbf{R}$ and the translation t up to a scale) can be determined. Finally,

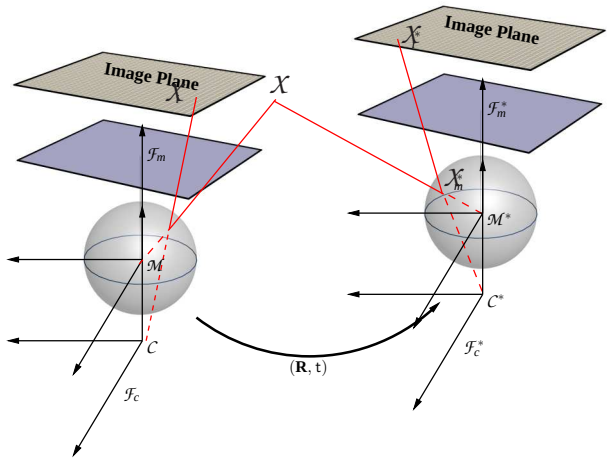


Fig. 5: Geometry of two views.

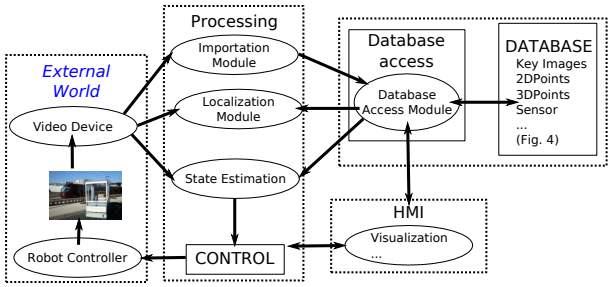


Fig. 6: Architecture of our software for visual navigation SoViN

the estimation of the input of the control law (3), *i.e* the angular deviation θ and the lateral deviation y, are computed straightforwardly from $\mathbf{R}$ and t.

IV. Implementation and Experimental Results

A. Map management

We have proposed a software platform called SoViN to efficiently manage visual memory for autonomous vehicle navigation in large scale environments [11]. An overview of SoViN is shown in Fig. 6. The software platform is divided into three different modules: a module for processing (image processing, computer vision and control); a module for HMI (visualization and high-level actions control) and a module for data storage and access (low-level functions). For data storage, our software uses a conventional database software. HMI and processing modules communicate with the database thanks to this low-level module.

During both localization and path following stages, key images' elements (image points with their descriptors, matching between successive image points ...) are loaded on-line from the database.

B. Experimental set-up

Our experimental vehicle is depicted in Figure 7. It is an urban electric vehicle, named RobuCab, manufactured by the Robosoft Company. Currently, RobuCab serves as experimental testbed in several French laboratories. The 4 DC motors are powered by lead-acid batteries, providing 2 hours autonomy. Vision and guidance algorithms are implemented in C^{++} language on a laptop using RTAI-Linux OS with a 2GHz Centrino processor. The Fujinon fisheye lens, mounted onto a Marlin F131B camera, has a field-of-view of 185 deg. The image resolution in the experiments was 800×600 pixels. It has been calibrated using the Matlab toolbox presented in [12]. The camera, looking forward, is situated at approximately 80cm from the ground. The parameters of the rigid transformation between the camera and the robot control frames are roughly estimated. Grey level images are acquired at a rate of 15fps.

Fig. 7: RobuCab vehicle with the embedded camera.

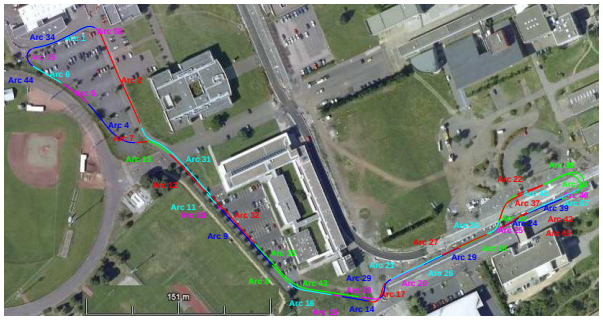


Fig. 8: Large-scale environment: large loop

C. Experimentations

1) Learning stage: The robot has been manually driven along a 1200 meter-long loop (refer to Fig. 8) at the beginning of July, with a very sunny weather. The fisheye lens camera was rigidly fixed at approximately 1 m from the ground, 1 m at the left of the middle of the car. The camera was looking in the direction of the vehicle. It has been calibrated using the unified model on the sphere. An importation step occurred and result to 35 edges after having cut some edges in function of the context (straight lines, huge turns). A longitudinal velocity has been given for each edge (0.4 m/s for huge turns, 0.8 m/s in small turns, 1 m/s for straight parts). The DGPS data has also been acquired. For each node, the position given by interpolation of the DGPS data have been saved too (interpolation in function of the time when data where acquired).

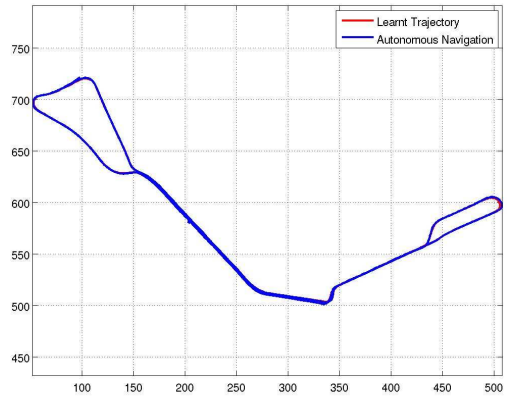


Fig. 11: Trajectories obtained with the RTK-DGPS

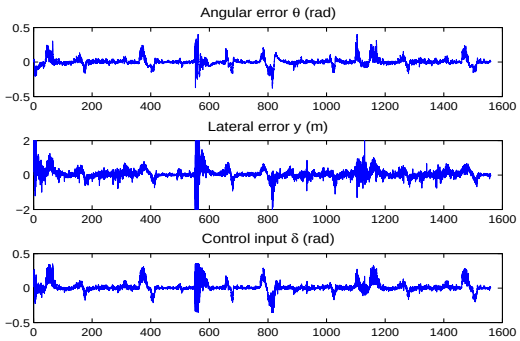


Fig. 9: Lateral y and angular θ errors and control input δ vs time (s).

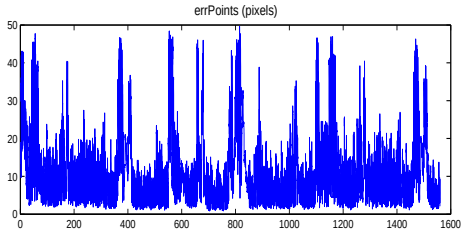


Fig. 10: Errors in the images (pixels) versus time (s).

2) Localization step and initialisation: After a localization step, a visual route is extracted. It consists on doing at least one loop.

3) Autonomous navigation: The experiment lasts 26 minutes for a path of 1700 meters (refer to Fig. 9) which results to an average longitudinal velocity of around 1 m/s. This visual route is composed of 54 edges and around 1400 key images. This path following stops for safety reasons because few visual features were robustly matched.

Evaluation with a RTK-GPS: DGPS data have been recorded during the learning and the autonomous stages. The results are reported in Fig. 11. The red and blue lines represents respectively the trajectories recorded during the learning and autonomous stages. It can be observed that these trajectories are similar. The lateral error measured by the RTK-GPS has a mean of 23 cm and a standard deviation of around 30 cm.

V. Conclusion

We have presented a complete framework for autonomous navigation which enables a vehicle to follow a sensory path obtained during a learning stage. The robot environment is modeled as a topological map from which a sensory route connecting the initial and goal images can be extracted. The robotic vehicle can then be driven along the route thanks to a sensor based control law which takes into account non-holonomic constraints. Furthermore, the state of the robot is estimated using a generic camera model valid for a perspective, catadioptric as well as a large class of fisheye cameras. Our approach has been validated on an urban vehicle navigating along a long trajectory. At our knowledge, it reports it is the first time that a 1700-meter-long trajectory is done using a single camera and natural landmarks.

Acknowledgment

This work is supported by the EU-Project FP6 IST μDrones, FP6-2005-IST-6-045248.

References

[1] P. Bonnifait, M. Jabbour, and V. Cherfaoui, "Autonomous Navigation in Urban Areas using GIS-Managed Information," *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1/2, pp. 83–103, 2008, special Issue on Advances in Autonomous Vehicles and Intelligent Transportation.

[2] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 1, Minneapolis, Minnesota, April 1996, pp. 83–88.

[3] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, february 2002.

[4] A. Diosi, A. Remazeilles, S. Segvic, and F. Chaumette, "Outdoor visual path following experiments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'07*, San Diego, CA, USA, 29 October - 2 November 2007, pp. 4265–4270.

[5] T. Zodiac, *Theory of robot control*, C. C. de Wit, B. Siciliano, and G. Bastin, Eds. Springer-Verlag, Berlin, 1995.

[6] C. Samson, "Control of chained systems. Application to path following and time-varying stabilization of mobile robots," *IEEE Transactions on Automatic Control*, vol. 40, no. 1, pp. 64–77, 1995.

[7] B. Thuilot, J. Bom, F. Marmoiton, and P. Martinet, "Accurate automatic guidance of an urban electric vehicle relying on a kinematic GPS sensor," in *5th IFAC Symposium on Intelligent Autonomous Vehicles, IAV'04*, Instituto Superior Técnico, Lisbon, Portugal, July 5-7th 2004.

[8] C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical implications," in *European Conference on Computer Vision*, vol. 29 (3), Dublin, Ireland, May 2000, pp. 159–179.

[9] T. Svoboda and T. Pajdla, "Epipolar geometry for central catadioptric cameras," *International Journal of Computer Vision*, vol. 49, no. 1, pp. 23–37, 2002.

[10] D. Nistér, "An efficient solution to the five-point relative pose problem," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[11] J. Courbon, L. Lequievre, Y. Mezouar, and L. Eck, "Navigation of urban vehicle: An efficient visual memory management for large scale environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'08*, Nice, France, sep 2008.

[12] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *IEEE International Conference on Robotics and Automation, ICRA'07*, Rome, Italy, apr 2007, pp. 3945–3950.

Session I

Vision based perception & Visual SLAM

- **Title: Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments**
  Authors: Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci, Domenico G. Sorrenti

Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments

Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci, Domenico G. Sorrenti

***Abstract*— The aim of this work is to demonstrate that it is possible to use a single camera to solve the problem of Simultaneous Localization And Mapping in dynamic environments obtaining, at the same time, the estimation of the moving objects trajectories. Specifically, we show that it is possible to segment the features belonging to independently moving objects from a moving camera using a MonoSLAM algorithm together with a Bearing-Only Tracker. The idea is to exchange between two parallel working systems, i.e. the SLAM filter and the bearing-only tracker, information about the pose of the camera and the motion of the feature to improve the robustness of the SLAM algorithm and maintain a consistent estimation of both the pose, the map, and the features trajectories. Experiments in simulated and real environments substantiate that the proposed technique is able to maintain consistent estimations in a fast and robust way suitable for a real-time application, even in situations where classical MonoSLAM algorithms are deemed to fail.**

I. INTRODUCTION

The key prerequisite for a complete autonomous navigation system is a deep understanding of the surrounding world as perceived by robot sensors. In Simultaneous Localization And Mapping (SLAM) literature it is possible to find many solutions using different kind of sensors (i.e. lasers, cameras, sonars) [1], but most of these algorithms assume a static environment or filter out the dynamic elements perceived in the scene [2].

Although the proposed approaches are effective, they are often expensive or complex and not usable for real applications. For this reason, in this paper, we focus on solutions based on a single camera, a small and inexpensive device that allows to have rich information about the environment perceived. In the last years we assisted the proliferation [3] of systems based on a single camera that are able to simultaneously localize themselves in real-time [4], building 3D maps of huge environments [5] and placing virtual elements in the scene [6]. However, as their precursors, they assume again a static environment.

In this paper we want to demonstrate that it is possible to relax the world motionless hypothesis, proposing a method to estimate online the 6 DoF of a camera and the 3D map in presence of generic dynamic objects.

A first remarkable work on this direction was done by Wang et al. [7], who proposed a mathematical framework to solve the problem of Simultaneous Localization And Mapping and Moving Objects Tracking (SLAMMOT), that can be considered the intersection between SLAM and moving object tracking. The authors investigate theoretically the SLAMMOT problem, demonstrating that it is possible to solve it maintaining separate posteriors for stationary and moving objects, and validating the algorithm empirically by analyzing data acquired with a laser rangefinder in real urban environment.

A different approach was presented by Bibby and Reid [8], introducing a technique called SLAMIDE, to combine the least-squares formulation of SLAM and sliding window optimization, together with a generalized expectation maximization method. Their idea is to incorporate both dynamic and stationary objects into SLAM estimation, without splitting the problem in two and considering the possibility of a reversible data association. Simulated experiments demonstrated the capabilities of the proposed solution, which is able to estimate, consistently, the pose and the map also in presence of dynamic features in a unique framework. However, as already demonstrated by Wang [7], the idea of including all the features in the SLAM state reduces the performance of the filter in terms of speed, highlighting the principal drawback of SLAMIDE: the complexity.

A different approach was proposed by Ess et al. [9], who presented a mobile system based on a stereo camera which integrates continuous visual odometry computation with tracking-by-detection, to track pedestrians in spite of frequent occlusions and egomotion of the camera rig. This method obtains interesting results in very challenging scenarios, but it is not a generic solution since it considers only pedestrian/vehicle tracking, and it is not computationally feasible for a robotics application. Moreover, no map is built, since the system is based on a visual odometry, thus it is not possible to have enough information about the environment to allow trajectory planning for an autonomous vehicle.

An approach requiring less computational resources, but still using a stereo camera, was introduced by Solà et al. [10], who described a system based on a framework called BiCamSLAM, that combines the advantages of the monocular reconstruction with the advantages of stereo vision. In his proposal, Solà tries to solve the SLAMMOT problem estimating, at the same time, the position of the robot, the static map and the trajectory of the moving objects. In particular, Solà proposes to separate the SLAM algorithm from the tracking one, adopting a camera-centric representation of the world and using a different filter for each moving object, dropping in this way objects crosscorrelations with

D. Marzorati and D. G. Sorrenti are with Università di Milano - Bicocca, Building U14, v.le Sarca 336, 20126, Milano, Italy{marzorati, sorrenti}@disco.unimib.it

M. Matteucci, R. Rigamonti and D. Migliore are with Politecnico di Milano, via Ponzio 34/5, 20133, Milano, Italy{matteucci, migliore}@elet.polimi.it

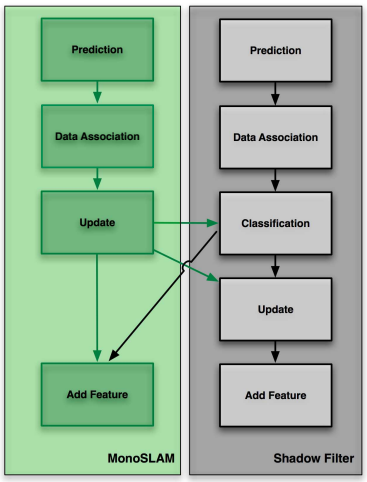


Fig. 1. Schema of the proposed SLAMBOT system. On the left we have the SLAM filter that, as explained in [11], estimates the camera pose and the position of the static elements in the scene by means of the EKF prediction, data association and update steps. The last pose estimated by the SLAM filter is then used by the the Shadow filter to identify dynamic features and estimate their trajectories. Once a feature is classified as static, it is added to the SLAM filter.

the robot's pose.

In this paper, starting from the Solà idea, a viable solution to the online monocular SLAM with moving objects tracking is proposed. The goal of our method is to obtain a consistent map of the static environment, discriminating between static and dynamic objects and being able at the meantime to approximate the trajectories of the moving features.

II. MONOSLAM WITH MOVING OBJECTS

Simultaneous estimation of pose and map based on the analysis of images perceived by a moving observer is not a trivial task; especially when the environment monitored contains dynamic elements that might affect the consistency of the estimates, leading to failure in the traditional SLAM algorithm. In the monocular case, this hindrance is worsened by the reconstruction procedure that is often unable to detect the dynamic behavior of a feature because of the high initial uncertainty associated with it [4].

A possible solution was proposed by Wang et al. [7], under the assumption that moving objects do not carry information about the map and the robot pose: he did not consider them as references for localization because of their inherent instability [10]. Exploiting this insight, we decided to split the estimation process over two filters reciprocally related (see Figure 1): the SLAM filter based on monocular camera (MonoSLAM), that uses static features to estimate map and camera pose, and the tracker, named in this paper "Shadow Filter", that, by knowing the camera pose, deals with the moving features in the environment. The role of the Shadow Filter is twofold: on one side, it tracks the behavior of the moving features, on the other, it retains the new features detected by the camera until it can tag them as static or dynamic, avoiding in this way inconsistencies in the SLAM process.

The system we propose relies on two main assumptions. Since we do not have any odometry measurement (i.e., we do not have an IMU), we need an absolute reference to understand how the camera and the feature are moving. Therefore, before perceiving dynamic features, we initialize the SLAM filter with a set of static features in known position (to estimate the scale), obtaining a first estimation of the camera pose w.r.t. the world frame. Moreover, to ensure consistent estimation and correct features classification during the whole execution of this system, it is important to have in the image and in the SLAM filter state enough static features to maintain an estimation of the absolute reference frame.

Under these assumptions, that could be easily relaxed by the use of an Inertial Measurement Unit (IMU), new features are initialized in the Shadow filter only. To avoid the corruption of the SLAM filter, these features are retained in it until it is not possible to mark them as static, in which case they are passed to the SLAM filter, or dynamic, in which case they are kept in the Shadow filter and tracked along their movements.

The MonoSLAM algorithm used in this work is the same proposed by Marzorati et al. [11], thus we avoid to explain here how this algorithm works, focusing, instead, on the description of the Shadow Filter side of the system and its interaction with the SLAM filter. However, it is simple to notice that the method proposed is independent of the SLAM algorithm used, since the only information exchanged are the camera pose and the feature positions.

III. DYNAMIC FEATURES TRACKING

As explained before, we propose to use a Bearing Only Tracker, the "Shadow Filter", to estimate and classify the new features perceived. Once we know the camera pose from the SLAM filter, to estimate the position and the velocity of a moving feature w.r.t. the camera frame we can use an EKF characterized by the following state:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{F_k}^{C_k} \\ \mathbf{v}_{F_k}^{F_k} \end{bmatrix}, \tag{1}$$

where $\mathbf{x}_{F_k}^{C_k} = (x_f, y_f, z_f, w_f)^T$ are the feature homogeneous coordinates at time k w.r.t. the camera frame C_k and $\mathbf{v}_{F_k}^{F_k} = (v_{f_x}, v_{f_y}, v_{f_z})^T$ is its velocity w.r.t. the feature frame F_k.

At each step we have to maintain the reference of the Shadow filter always w.r.t. the camera frame, thus we need to roto-translate the feature position and rotate the velocity vector before the update step. Assuming constant velocity, we can write the motion equation as:

$$\mathbf{x}_{k+1} = \begin{bmatrix} \mathbf{x}_{F_{k+1}}^{C_{k+1}} \\ \mathbf{v}_{F_{k+1}}^{F_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{C_k}^{C_{k+1}} \oplus \mathbf{x}_{F_k}^{C_k} \oplus (\mathbf{v}_{F_{k+1}}^{F_k} \Delta t) \\ \mathbf{x}_{F_k}^{F_{k+1}} \oplus \mathbf{v}_{F_k}^{F_k} \end{bmatrix}, \tag{2}$$

where: $\mathbf{x}_{C_k}^{C_{k+1}}$ is the roto-translation between the camera poses C_k and C_{k+1}, $\mathbf{x}_{F_k}^{C_k}$ is the feature position w.r.t. the camera pose at time k, $\mathbf{v}_{F_{k+1}}^{F_k}$ is the velocity of the feature

at time $k+1$ w.r.t. the feature frame F_k, $\mathbf{v}_{F_k}^{F_k}$ the velocity of the feature at time k w.r.t. the feature frame F_k, $\mathbf{x}_{F_k}^{F_{k+1}}$ is the rotation from the frame reference at time k to the frame reference at time $k+1$ and $\oplus$ is the transformation composition operator. Notice that the state of the feature is somehow represented in a mixed frame of reference to simplify the motion model: its position is in the camera frame, while its velocity is in the feature frame (i.e., the camera reference translated in the feature point).

The measurement equation in homogeneous coordinates can be written as:

$$\mathbf{h}_k = \begin{bmatrix} h_{k_x} \\ h_{k_y} \\ h_{k_z} \end{bmatrix} = \mathbf{M}\mathbf{x}_{F_k}^{C_k}, \tag{3}$$

where $\mathbf{M}$ is the calibration matrix:

$$\mathbf{M} = \begin{bmatrix} f_{c_x} & 0 & cc_x \\ 0 & f_{c_y} & cc_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

and the pixel coordinates on the image plane can be simply obtained as

$$\mathbf{h}_k = \begin{bmatrix} h_{k_u} \\ h_{k_v} \end{bmatrix} \begin{bmatrix} h_{k_x}/h_{k_z} \\ h_{k_y}/h_{k_z} \end{bmatrix}. \tag{5}$$

For the experiments shown in this paper we used a camera with a wide-angle lens, to improve the performance of single-camera SLAM [12], thus the measurement equation should be modified accordingly to take into account the radial distortion, as exposed in [11]. Finally, to estimate iteratively the current position[1] of the feature, we just need to compute the Jacobian of these models and apply the classical steps of the Extedend Kalman Filter.

This approach allows us to have an approximated estimation of the feature pose and, in this way, make inference about its movements.

A. Detecting moving features

To guarantee the correct functioning of the SLAM algorithm, we need to classify new features as dynamic or static before using them to estimate the camera pose and the map. The first time we perceive a feature, we do not know where it is located in the 3D scene, thus we initialize it with a huge uncertainty in the depth. In the next frame, once the feature is associated with a measurement and then updated, its position changes, moving along the projection ray and possibly causing the estimate of false motion. For this reason we can not rely the velocities estimated in the Shadow Filter and we need a more robust classifier.

Referring to the viewing ray as a straight line and to the position from where the feature was viewed the first time, we can make a geometric reasoning, based on an approach that resembles the epipolar constraint. The basic idea is to check continuously the intersections between three viewing rays belonging to the same feature viewed in three different camera poses. If these intersections are not the same during the camera motion or it does not exist, then the feature can be classified as dynamic.

However, in real world, where the moving sensor returns uncertain bearing-only measurements, the previous task is not trivial to solve, since the presence of the uncertainty could affect all the geometric reasoning. To take into account the uncertainty associated with each measurement and each estimate, we need to introduce a probabilistic framework that allows us to check the relationships between the viewing rays in an uncertain world: Uncertain Projective Geometry [15].

Using this framework we can describe, combine, and estimate various types of geometric elements (3D points, 3D lines and 3D planes) maintaing the information about their uncertainty. By the use of Uncertain Projective Geometry, these elements are represented using homogeneous vectors (using the Plücker coordinates for lines) with their covariance matrices, and simple bilinear expressions to represent join and intersection operators are used. This result can be obtained by using two construction matrices: $\mathbf{O}(\cdot)$ (for 3D lines) and $\mathbf{\Pi}(\cdot)$ (for 3D points and 3D planes).

To join two 3D points $\mathbf{X} = (X_1, Y_1, Z_1, W_1)^T$, $\mathbf{Y} = (X_2, Y_2, Z_2, W_2)^T$ into a 3D line $\mathbf{L}$ expressed in Plücker coordinates [15], we can write:

$$\mathbf{L} = \mathbf{X} \wedge \mathbf{Y} = \mathbf{\Pi}(\mathbf{X})\mathbf{Y}, \tag{6}$$

being

$$\mathbf{\Pi}(\mathbf{X}) = \frac{\partial \mathbf{X} \wedge \partial \mathbf{Y}}{\partial \mathbf{Y}} = \begin{pmatrix} W_1 & 0 & 0 & -X_1 \\ 0 & W_1 & 0 & -Y_1 \\ 0 & 0 & W_1 & -Z_1 \\ 0 & -Z_1 & Y_1 & 0 \\ Z_1 & 0 & -X_1 & 0 \\ -Y_1 & X_1 & 0 & 0 \end{pmatrix}. \tag{7}$$

Again we can join a 3D point $\mathbf{X} = (X_1, Y_1, Z_1, W_1)^T$ with a 3D line $\mathbf{L} = (L_1, L_2, L_3, L_4, L_5, L_6)$ into a 3D plane $\mathbf{A}$:

$$\mathbf{A} = \mathbf{X} \wedge \mathbf{L} = \mathbf{O}(\mathbf{L})\mathbf{X}, \tag{8}$$

$$\mathbf{O}(\mathbf{L}) = \frac{\partial \mathbf{X} \wedge \partial \mathbf{L}}{\partial \mathbf{X}} = \begin{pmatrix} 0 & L_3 & -L_2 & -L_4 \\ -L_3 & 0 & L_1 & -L_5 \\ L_2 & -L_1 & 0 & -L_6 \\ L_4 & L_5 & L_6 & 0 \end{pmatrix}. \tag{9}$$

These construction matrices are useful tools to derive new geometric entities from known ones, e.g. a 3D line from two 3D points, a 3D point from the intersection of two 3D lines, etc.; at the same time, being bilinear equations, these operators directly represent the Jacobian of the transformation which is used for the uncertainty propagation in the construction process.

A new entity z can be estimated from two entities x and y, with a simple matrix multiplication:

$$z = f(x, y) = U(x)y = V(y)x, \tag{10}$$

where $U(x)$ and $V(y)$ are, at the same time, the bilinear operators and the Jacobian of the x and y entity respectively.

[1]Notice that this filter can estimate the trajectories of the moving points up to a scale factor [13], however it is possible to overcome this drawback initializing the correct scale in the first frame, as showed in [14].

Assuming the entities to be uncertain, the pairs (x, Σ_{xx}) and (y, Σ_{yy}), and possibly the covariances Σ_{xy} between x and y, are required for computing the error propagation as:

$$(z, \Sigma_{zz}) = \left(U(x)y, [V(y), U(x)] \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{xy} & \Sigma_{yy} \end{pmatrix} \begin{bmatrix} V^T(y) \\ U^T(x) \end{bmatrix} \right), \quad (11)$$

and in case of independence between x and y we obtain:

$$(z, \Sigma_{zz}) = \left(U(x)y, U(x)\Sigma_{yy}U^T(x) + V(y)\Sigma_{xx}V^T(y) \right). \quad (12)$$

To check the geometric relationship between two geometric entities it is then possible to use a statistical test on the distance vector d defined using the previous bilinear equation. In particular a relation can be assumed to hold if the hypothesis

$$H_0 : d = U(x)y = V(y)x = 0 \quad (13)$$

cannot be rejected. Notice that the hypothesis H_0 can be rejected with a significance level of α if

$$T = d^T \Sigma_{dd}^{-1} d > \varepsilon_H = \chi^2_{1-\alpha;n} \quad (14)$$

To perform the test, we need to fix the probability α that we reject H_0 although it is actually true and this situation is called Type-I error. The probability α is usually a small number such as 1% or 5% and it is called significance level of the test. The critical value ε_H such that $P(T > \varepsilon_H | H_0) = \alpha$ is given by the $(1-\alpha)$-quantile of the χ^2 distribution. It is crucial to note that a successful hypothesis test $T < \varepsilon_H$ does not validate that H_0 is true, it merely states that there is not enough evidence to reject H_0.

The covariance matrix Σ_{dd} of d is given by first order error propagation as

$$\Sigma_{dd} = U(x)\Sigma_{yy}U^T(x) + V(y)\Sigma_{xx}V^T(y)$$

In general Σ_{dd} may be singular, if d is a n x 1 vector, r is is the degree of freedom of the relation R and $r < n$. The singularity causes a problem, as we have to invert the covariance matrix. But, at least for projective relations, it can be guaranteed that the rank of Σ_{dd} is not less than r (see Heuel [15] for more details).

IV. EXPERIMENTAL RESULTS

In this section we want to test the capabilities of our system, verifying the result of dynamic classification and the consistence of the estimated position and map. Before trying the algorithm with real data, we verified the consistency of the Shadow filter, testing it in a simulated framework, in which a moving camera was put inside an environment where another dynamic element is moving in the scene (see Figure 2 for a reference). At each time the correct camera position is passed to the Shadow filter and the trajectory of the feature is estimated. As it is possible to notice from Figure 3, the estimate remains consistent during the whole process. The uncertainty associated to the depth coordinate (in the case of the experiment this can be identified with the X coordinate) is higher than the uncertainties associated to the other coordinates, making the Shadow filter estimates

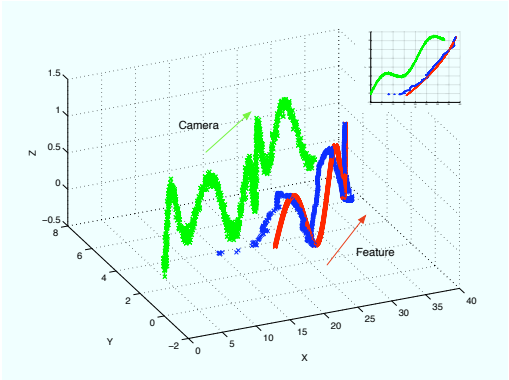


Fig. 2. In this image we show the trajectory of the camera (in green) and of the feature (in red), simulated to test the capabilities of the Shadow filter. In blue it is possible to see the accuracy of the estimated position (the small image represents the projection of the same scene on the XY plane).

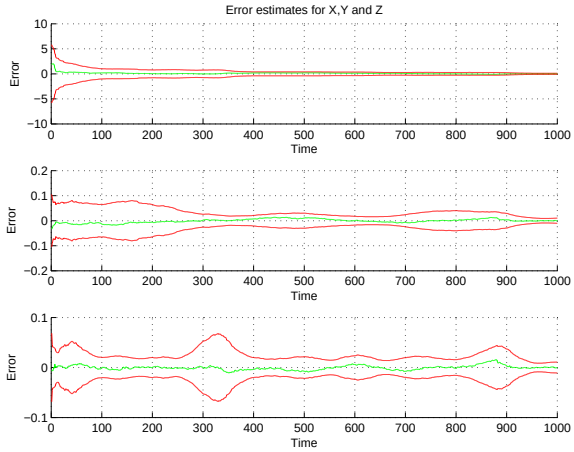


Fig. 3. Consistency test for the Shadow filter. In the plots we show the estimation error (in green) for the x, y, z coordinates of the feature respectively. In red we present the $\pm 3\sigma$ threshold for the covariance; notice that the errors are always contained between those bounds, thus this filter remains consistent.

unfeasible for accurate tracking. This drawback is principally due to consecutive violations of the observability conditions. In fact the displacements between two consecutive steps are so small to cause the partial unobservability of the homogenous part of the feature and a consequent increase of the uncertainty associated to the depth component. This simple analysis gives us information about the quality and the accuracy of the estimates, but also provides an important insight: the observability condition can be easily violated in an online MonoSLAM application.

Although we can not localize accurately the moving object, the consistency of the filter demonstrates the validity of the reasoning based on the uncertainty geometry approach (notice that the errors is always included in the $\pm 3\sigma$ uncertainty interval) and it proves that, taking into account the estimate uncertainty, we can robustly classify a feature as static or dynamic.

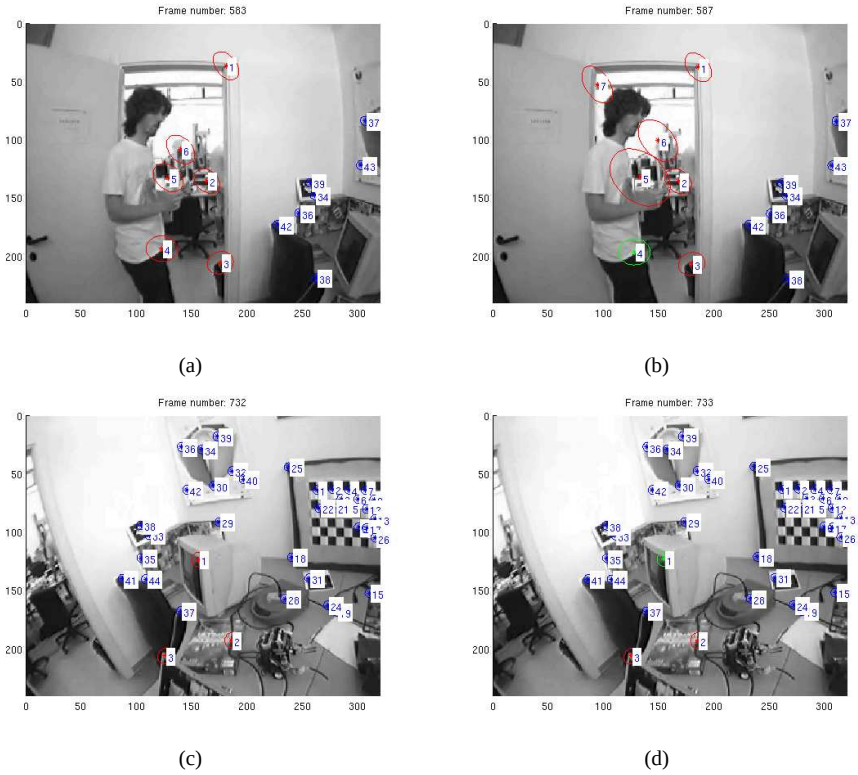


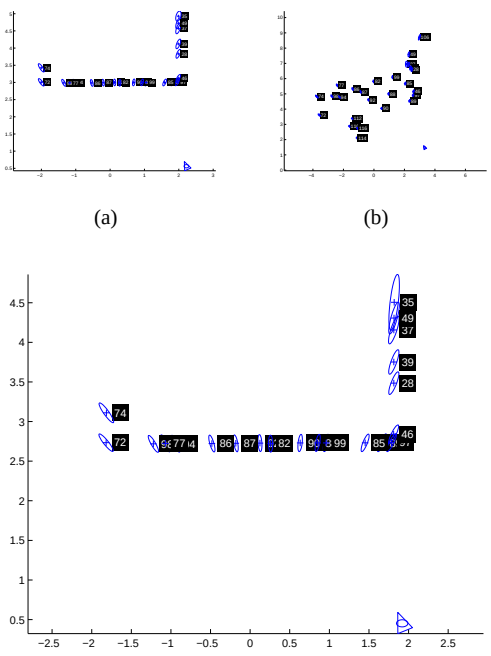
Fig. 4. Static/Dynamic classifier results. In the first row it is possible to see an example of dynamic (in green), static (in blue) classification. The features in the Shadow filter that are waiting for a classification are showed in red. In the last row we can see a feature erroneously classified as moving. This kind of error is expected since the classification is based on a probabilistic test with a threshold of 95%.

This statement can be validated by testing the classifier algorithm on real datasets. In Figure 4 it is possible to see two examples representing both a correct and a wrong classification. We have tested the algorithm using many real datasets and we noticed that, if the feature is correctly matched, the algorithm always distinguish between moving and static features. Sometimes it is possible to have a static feature classified as dynamic (see again Figure 4(c) 4(d)), but it never happened to confuse a moving feature as static. Albeit the probabilistic test has an expected failure rate of the 5%, this contingency happened rarely in our experiments (see again Figure 4(c) 4(d) for an example) and, since it does not corrupt the SLAM filter, it can be tolerated.

Finally we were interested in verifying that our system is able to improve the estimates quality when there are dynamic features in the environment. For this purpose we set up a simulated 3D environment characterized by features both static and dynamic. Data association was performed manually to avoid possible errors due to mismatches and to evaluate the quality of the pose and of the estimated map against a ground truth. In Figure 5 it is possible to see the improvements carried by the use of the Shadow Filter. In the first plot (Figure 5(a)) it is possible to see the map resulting from the use of the classic MonoSLAM algorithm using only the static features. In Figure 5(b) it is shown the results using always the classic MonoSLAM, but this time introducing the dynamic elements, and in the last image (Figure 5(c)) the resulting map obtained introducing the Shadow filter. It is also possible to see how a traditional SLAM filter, that does not identify and exclude from estimation the dynamic features, introduces a set of errors that lead to failure. If we correctly identify the dynamic features, we can avoid to initialize them inside the SLAM filter, maintaining the same accuracy of a SLAM system operating in a purely-static environment. In Figure 6 it is possible to see the results obtained using the real dataset. Despite the presence of dynamic features that could affect the SLAM algorithm, the estimated map remains consistent and, when the camera perceives again the checkerboard, the features are re-matched correctly, closing the loop.

V. CONCLUSIONS AND FUTURE WORKS

In this paper we have proposed a novel solution for the problem of Simultaneous Localization, Mapping and Moving Object Tracking, when using a single camera as a sensor. To avoid errors in the SLAM estimates, we demonstrated that it is possible to identify online the static and dynamic features, using an approach based on the Uncertain Geometry proposed by Heuel [15], that allows to detect the moving features with a simple statistical test. The experimental results confirmed the capabilities of this approach that can be used online in real application and, potentially, with any MonoSLAM algorithm with performances that allow online execution, since it does not require any particular modification of the original SLAM algorithm.



(a) (b) (c)

Fig. 5. In this image we show the estimated map when we have an environment containing moving feature, using the MonoSLAM proposed in [11] (b) and using the MonoSLAMBOT approach (c). This result can be compared with the map obtained using the MonoSLAM and "disabling" the dynamic features (a).

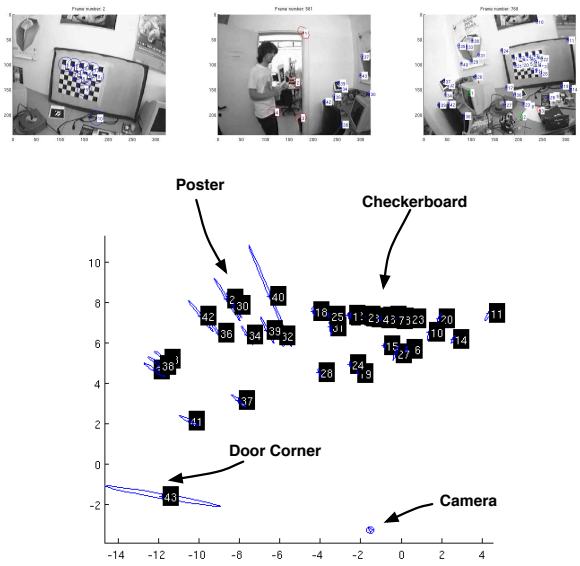


Fig. 6. The resulting map obtained using the MonoSLAMBOT approach is robust to moving elements in the scene (in this case a person). The estimates are consistent and allow the loop closure on the checkerboard.

One limitation of this work is due to the difficulty of tracking robustly the moving elements between frames at different time (e.g., the interest points detected on a walking person, as in Figure 6, could change considerably during the acquisition). For this reason it is not always possible to reach the convergence of the Shadow Filter and, as a consequence, to obtain an accurate estimate of the moving objects in the scene. In the future we want to cope with this limitation introducing an analysis of the structure of the scene, e.g., clustering points with similar dynamics [16] or adopting a Tracking-by-Detection approach [17], to introduce enough constrains to reduce the uncertainty associated with each point. Moreover we plan to investigate a possible extension based on the integration with an IMU to remove the constraints over the first frame, the need to perceive enough static features in each image frame, and to allow a direct estimate of the real scale.

VI. ACKNOWLEDGMENTS

This work has been partially supported by the European Commission, Sixth Framework Programme, Information Society Technologies: Contract Number FP6-045144 (RAWSEEDS), and by Italian Institute of Technology (IIT) grant.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, June 2006.

[2] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *IEEE ICRA*, 2003.

[3] J. Neira, A. Davison, and J. Leonard, "Guest editorial, special issue in visual slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 929–931, October 2008.

[4] J. Montiel, J. Civera, and A. Davison, "Unified inverse depth parametrization for monocular slam," in *RSS*, 2006 2006.

[5] L. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: Ekf slam in o(n)," *IEEE Transactions on Robotics*, p. (to appear), 2008.

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *ACM ISMAR*, 2007.

[7] C.-C. Wang, "Simultaneous localization, mapping and moving object tracking," Ph.D. dissertation, Robotics Institute Carnegie Mellon University, 2004.

[8] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association," in *Proceedings of RSS*, 2007.

[9] A. Ess, B. Leibe, K. Schindler, and L. van Gool, "A mobile vision system for robust multi-person tracking," in *IEEE CVPR*. IEEE Press, June 2008.

[10] J. Sola, "Towards visual localization, mapping and moving objects tracking by a mobile robot: a geometric and probabilistic approach," Ph.D. dissertation, Institut National Politechnique de Toulouse, 2007.

[11] D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti, "Monocular slam with inverse scaling parametrization," in *BMVC*, 2008, pp. 945–954.

[12] A. Davison, Y. G. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," in *Proc. IFAC Symposium on IAV*, Jul. 2004.

[13] R. Vidal, S. Soatto, and S. Sastry, "A factorization method for 3d multi-body motion estimation and segmentation," in *ACSS*, October 2002, pp. 1637–1646.

[14] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on PAMI*, vol. 29, no. 6, pp. 1052–1067, 2007.

[15] S. Heuel, *Uncertain Projective Geometry: Statistical Reasoning for Polyhedral Object Reconstruction*. Springer, 2004.

[16] K. E. Ozden, K. Schindler, and L. van Gool, "Simultaneous segmentation and 3d reconstruction of monocular image sequences," in *ICCV*, October 2007.

[17] B. Leibe, K. Schindler, and L. J. V. Gool, "Coupled detection and trajectory estimation for multi-object tracking," in *ICCV*, 2007, pp. 1–8.

ICRA2009

2009 IEEE International Conference
on Robotics and Automation
Kobe, Japan, May 12-17, 2009

Session I

Vision based perception & Visual SLAM

- **Title: Optimal Metric SLAM for Autonomous Navigation Assistance**
  Authors: P.F. Alcantarilla, I. Parra, L.M. Bergasa

Optimal Metric SLAM for Autonomous Navigation Assistance

P.F. Alcantarilla, L.M. Bergasa, I. Parra, D. Schleicher
Department of Electronics. University of Alcalá
Alcalá de Henares (Madrid), Spain
pablo.alcantarilla, bergasa, parra@depeca.uah.es, dsg68818@telefonica.net

***Abstract*— In this paper we present a 6DOF metric SLAM system for outdoor enviroments using a stereo camera, mounted next to the rear view mirror, as the only sensor. By means of SLAM the vehicle motion trajectory and a sparse map of natural landmarks are both estimated at the same time. The system combines both bearing and depth information using two different types of feature parametrization: inverse depth and 3D. Through this approach near and far features can be mapped, providing orientation and depth information respectively. Natural landmarks are extracted from the image and are stored as 3D or inverse depth points, depending on a depth threshold. At the moment each landmark is initialized, the normal of the patch surface is computed using the information of the stereo pair. In order to improve long-term tracking a 2D warping is done considering the normal vector information of each patch. This Visual SLAM system is focused on the localization of a vehicle in outdoor urban environments and can be fused with other cheap sensors such as GPS, so as to produce accurate estimations of vehicle's localization in a road. Some experimental results under outdoor environments and conclusions are presented.**

I. Introduction

Real-time Simultaneous Localization and Mapping has an important key role in robotics. In recent times, SLAM has captured the attention of computer vision researchers and the interest of using cameras as sensors has grown considerably due to mainly three reasons. Cameras are cheaper than commonly used scan-lasers, they provide rich visual information about scene elements and are easy to adapt for wearable systems. According to that, the range of SLAM based applications has spread to non typical robotic environments such as augmented reality [1], non-invasive surgery [2] and vehicle localization [3].

In this work a 6DOF Stereo SLAM system is proposed in order to develop a robust localization system, using only a cheap stereo camera mounted next to the rear view mirror, able to complement a standard GPS sensor for autonomous vehicle navigation where GPS signal does not exist or it is not reliable (tunnels, urban areas...). At the same time, a sparse map of high quality features is computed. This optimized map contributes to a better localization estimate and prevents the system from drifting in situations where the vehicle visits some areas that were previously visited, i.e. loop closing situations. The main advantages of using a stereo system instead of a monocular one were described in [4].

The traditional approach in in the literature for solving the SLAM problem, is using an extended Kalman Filter (EKF) with the vehicle pose and static landmarks as the evolving filter state. This EKF approach has some drawbacks as it is explained in [5]. The main drawback of the EKF implementation is that the computational requirement for the filter update increases quadratically in large-scale maps as a function of the landmarks introduced into the filter $O(n^2)$. A typical solution to cope with this problem is submapping, where the global map is obtained fusing the information from local submaps [3], [6].

Our system follows a Davison's SLAM approach [7]. That is, a few high quality features are tracked and used to compute the position of the camera creating a sparse map of high quality textured landmarks using an Extended Kalman Filter (EKF). Paz et al. proposed in [8] a 6DOF Stereo EKF-SLAM system with stereo in hand for large indoor and outdoor environments. The inverse depth parametrization proposed by Civera et al. [9] for the MonoSLAM approach is adapted to the StereoSLAM version so as to provide distance and orientation information. Point features are extracted from the images and are classified as 3D features if the disparity is enough, or stored as inverse depth features otherwise. Their Visual SLAM algorithm generates conditionally independent local maps and finally, the full map is obtained using the Conditionally Independent Divide and Conquer algorithm, which allows constant time operation most of the time [6]. Although results are good considering large maps in indoor/outdoor environments, the range of camera movements is limited, since no patch adaptation is done and only 2D image templates correlations are carried out in the matching process. By means of an empirical analysis, they suggest choosing a threshold of depth 5 m, for switching between inverse depth and 3D features. Besides, our sequences are more suitable to show the benefits of an inverse depth parametrization for far features.

The accuracy of the stereo sensor is limited up to a certain depth, depending mainly on the baseline of the sensor. In typical outdoor road vehicles sequences, is common to have very far landmarks. If we try to measure the 3D position of a far feature, which is located beyond the limits of our sensor, the uncertainty in the measurement will be very high. On the contrary we can

reduce the uncertainty of far features if we just measure the orientation of the feature.

The two key contributions of our work, are the use of inverse depth and 3D features for providing both depth and angular information, and a 2D homography warping method considering information from both cameras of the stereo pair. This paper is organized as follows: the general structure of the system is explained in section II. In section III we deal with the problem of how to switch between inverse depth or 3D parametrization. In The 2D homography warping for patch adaptation is described in section IV. Finally, some experimental results are presented in section V. Main conclusions and future works are discussed in section VI.

II. System Structure

Our system is based on a stereo camera mounted on a mobile vehicle close to the rear view mirror. Fig. 1 depicts the common type of sequences in outdoor road vehicle navigation. As it can be observed, some features are very far with respect to the camera, whereas we can have some features close to the camera. Both far and close features are displayed in orange (weak) and red (dark) respectively in Fig. 1.

Fig. 1. Typical outdoor road navigation sequences

The global state vector X incorporates the information for the left camera and for the features. The camera state is composed of its 3D position using cartesian coordinates, the camera orientation in terms of a quaternion, and the linear and angular speed, which are necessary for the impulse motion model used for modelling the camera movement. The motion model that is assumed is a constant velocity and constant angular velocity model explained in [7].

$$\mathbf{X_v}_{\,[\mathbf{13},\mathbf{1}]} = (X_{cam},\ q_{cam},\ v_{cam},\ \omega_{cam})^t \qquad (1)$$

Two types of feature parametrization are used providing orientation and depth information respectively. Depending on the depth of the feature as described in section III, features are initialized as inverse depth or 3D and are incorporated to the EKF SLAM algorithm.

$$\mathbf{X} = (X_v, Y_{1\ 3D} \cdots Y_{n\ 3D}, Y_{1\ INV} \cdots Y_{m\ INV})^t \qquad (2)$$

Interesting points are extracted from the image using the Harris corner detector [10] and a subsequent subpixel refinement. When the camera moves, these features are tracked over the time to update the filter. In order to track a feature, image position is predicted in both cameras. Then, the feature appearance is transformed using a 2D homography according to section IV, and a correlation search is performed inside a search area of high probability which is defined by the uncertainties of the feature and the camera. ZMCC (Zero Mean Cross Correlation) is used since its robustness against lighting changes. An intelligent feature management is implemented, so low-quality features are deleted from the state vector.

Due to the use of a wide-angle lens, it is necessary to use a distortion model correcting distorted images. Unlike other SLAM systems [4], [7] radial and tangential distortion are corrected using LUT (Look up tables), so images are corrected previous to processing. Two main advantages are obtained from using LUTs: firstly, this method is faster than working with the distorted images and then correcting the distorted projection coordinates, and secondly, the matching process is less critical if undistorted images are used.

A. 3D Features

For 3D features, the feature's state vector encodes the information about the 3D position of the feature in the global map reference system.

$$\mathbf{Y_{3D}}_{\,[\mathbf{3},\mathbf{1}]} = (x,\ y,\ z)^t \qquad (3)$$

B. Inverse depth Features

For inverse depth features, the feature's state vector encodes the information of the 3D optical center pose from which the feature was first seen X_{ori}, the orientation of the ray passing through the image point (angles of azimuth θ and elevation ϕ) and the inverse of its depth, ρ. Fig. 2 depicts the inverse depth point coding:

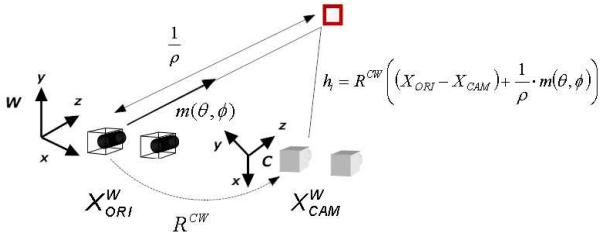


Fig. 2. Inverse depth point coding

$$\mathbf{Y_{INV}}_{\,[\mathbf{6},\mathbf{1}]} = (X_{ori},\ \theta,\ \phi,\ \rho)^t \qquad (4)$$

In Fig. 2, $m(\theta, \phi)$ is the unitary ray directional vector from the camera to the feature. This unitary vector is defined according to eq. 5:

$$\mathbf{m}(\theta,\phi)_{\ [\mathbf{3,1}]} = (\sin\phi\cos\theta, -\cos\phi, \sin\phi\sin\theta)^t \quad (5)$$

The angles of azimuth and elevation are defined as follows:

$$\theta = tan^{-1}\left(\frac{z}{x}\right) \quad (6)$$

$$\phi = tan^{-1}\left(\frac{\sqrt{x^2+z^2}}{y}\right) \quad (7)$$

III. Switching between Inverse Depth and 3D Features

Harris corners are extracted from the images and are classified as 3D features or stored as inverse depth features, depending on a depth threshold. This depth threshold is empirically set to 30 m. The value of this threshold is chosen as a compromise between non-linearity measurements, features uncertainty and the overhead introduced by the inverse depth parametrization. After some experimental tests we found the value of 30 m as a good threshold for our application.

Once the features are predicted in the EKF prediction step, it is necessary to determine if the original parametrization of the features has to be changed (i.e. if an inverse depth feature is now below the depth threshold and should adapt a 3D parametrization or viceversa). Besides, a constraint is imposed: the feature has to remain at least m frames (typically 15 frames) in its new parametrization state before the switching. This is done in order to avoid unnecessary switchings in case that the depth estimate is above and below the threshold in consecutive frames.

When an inverse depth feature is switched to a 3D parametrization, it is necessary to adapt the feature's state and the covariances implied in the filtering process by means of equations 8 for the feature's state and 8, 10 for the covariances. In the same way we can switch between 3D features to inverse depth, although this is not a common case in autonomous navigation.

$$\mathbf{Y_{3D}}_{\ [\mathbf{3,1}]} = X_{ORI} + \frac{1}{\rho}\cdot m(\theta,\phi) \quad (8)$$

$$\mathbf{P_{YY_{3D}}}_{\ [\mathbf{3,3}]} = \left(\frac{\partial Y_{3D}}{\partial Y_{INV}}\right)\cdot P_{YY_{INV}}\cdot\left(\frac{\partial Y_{3D}}{\partial Y_{INV}}\right)^t \quad (9)$$

$$\mathbf{P_{XY_{3D}}}_{\ [\mathbf{13,3}]} = P_{XY_{INV}}\cdot\left(\frac{\partial Y_{INV}}{\partial Y_{3D}}\right)^t \quad (10)$$

IV. 2D Homography Warping

When a feature is going to be measured, the estimation of the left camera position and orientation, which are obtained both from the SLAM state vector, and the normal surface patch vector are used for transforming the initial image template appearance (due to changes in viewpoint) by warping the initial template using a 2D homography. Our approach is related to the previous works of [11], [12].

Considering two camera centered coordinate systems, the transformation between two generic coordinate systems X_1 and X_2 is defined by:

$$X_2 = R\cdot X_1 + T \quad (11)$$

where R and T are the rotation matrix and the traslation vector encoding the relative position of the two coordinate systems. If X_1 is a point on the plane defined by eq. 12:

$$\pi : a\cdot x_1 + b\cdot y_1 + c\cdot z_1 + 1 = 0 \quad (12)$$

This is a plane which does not pass through the origin, and $n = (a,b,c)^t$ is the plane normal. According to this, the following relationship can be found:

$$n^t\cdot X_1 = -1 \quad (13)$$

Using the previous equation, eq. 11 can be expressed as follows:

$$X_2 = R\cdot X_1 - T\cdot n^t\cdot X_1 = \left(R - T\cdot n^t\right)\cdot X_1 \quad (14)$$

And therefore, image positions in the two camera frames are related by the 2D homography:

$$U_2 = C_2\cdot\left(R - T\cdot n^t\right)\cdot C_1^{-1}\cdot U_1 \quad (15)$$

Fig. 3 depicts the stereo geometry, and also the problems of obtaining the plane normal vector and the 2D homography for warping the initial image template using information from both cameras.

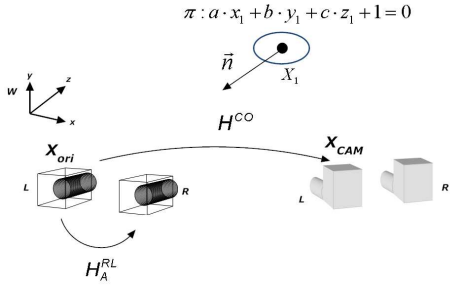


Fig. 3. Stereo geometry and locally planar surfaces

Eq. 16 denotes the relationship between the left camera and the right camera coordinate systems:

$$U_R = C_R\cdot\left(R^{RL} - T^{RL}\cdot n^t\right)\cdot C_L^{-1}\cdot U_L \quad (16)$$

The previous equation depends on the rotation matrix R^{RL} and the translation vector T^{RL} between both cameras. The values of these matrixes are known accurately, since they are estimated in a previous stereo calibration process. Supposing an affine transformation between left

and right image patches, the affine transformation H_A^{RL} can be expressed as:

$$\mathbf{H_A^{RL}} \;=\; C_R \cdot \left(R^{RL} - T^{RL} \cdot n^t\right) \cdot C_L^{-1} \qquad (17)$$

This affine transformation can be computed easily by means of 3 correspondences of non collinear points and with the assumption of locally planar patches. As it can be observed, eq. 17 depends on the plane normal vector n. From eq. 17 the product $T^{RL} \cdot n^t$ can be isolated. Denoting this product as X, it can be obtained as follows:

$$\mathbf{X} \;=\; T^{RL} \cdot n^t \;=\; R^{RL} - C_R^{-1} \cdot H_A^{RL} \cdot C_L \qquad (18)$$

All the parameters of eq. 18 are known, since the affine transformation H_A^{RL} has been previously computed, and the rest of implied matrixes are known from the stereo calibration process. According to this, a system of 9 equations and 3 unknowns, which are the components of the plane normal vector, can be found:

$$\begin{cases} n_x = \frac{X_{11}}{T_x} & n_x = \frac{X_{21}}{T_y} & n_x = \frac{X_{31}}{T_z} \\ n_y = \frac{X_{12}}{T_x} & n_y = \frac{X_{22}}{T_y} & n_y = \frac{X_{32}}{T_z} \\ n_z = \frac{X_{13}}{T_x} & n_z = \frac{X_{23}}{T_y} & n_z = \frac{X_{33}}{T_z} \end{cases} \qquad (19)$$

At the moment of a feature initialization, the plane normal vector is computed in the way it has been explained. Once this normal vector is estimated, the 2D homography between two different viewpoints can be determined using the estimation of the current left camera position and orientation and the left camera position and orientation: from the feature initialization viewpoint:

$$U_{CAM} \;=\; C_L \cdot \left(R^{CO} - T^{CO} \cdot n^t\right) \cdot C_L^{-1} \cdot U_{ORI} \qquad (20)$$

where R^{CO} and T^{CO} are the rotation and translation matrixes between the current left camera position and the reference position when the feature was initialized.

V. Experiments in Outdoor Environments

In order to test the system performance, lots of outdoor sequences in urban environment under real traffic condictions have been tested. In this work, we present only the results of two of them. The cameras used were the Unibrain Fire-i IEEE1394 modules with a baseline of 30 *cm*. Image resolution was 320×240 pixels and the images were B&W sequences. The acquisition frame rate was 30 frames per second. The sequences were processed on a laptop with an Intel Core 2 Duo processor at 2.4GHz. Camera calibration is done in a previous setup process. The Visual SLAM algorithm is implemented in C/C++ and works in real-time as long as the number of features doesn't exceed 150 approximately.

Figures 5(a) and 5(b) illustrate the aerial views of the trajectory done by the vehicle in each of the sequences. For each of the sequences two different simulations were done: without inverse depth parametrization (only 3D parametrization) and considering both parametrizations (inverse depth and 3D) with a depth threshold of 30 m.

The final map and trajectory for the first and second sequences are displayed in Fig. 6 and Fig. 7 respectively, considering the different cases. Table I shows the results of the comparison between the different experiments. The meaning of the parameters of this table are:

- % Inverse Features: Is the percentage of the total number of features in the map that were initialized with an inverse depth parametrization.
- Estimated Length (m): Is the estimate of the total distance covered by the vehicle in the sequence.
- Mean P_{YY} Trace: Is the mean trace of the covariance matrix P_{YY} for each of the features that compose the final map. This parameter is indicative of the uncertainty of the features, i.e. the quality of the map.

In the first experiment, the car starts turning slightly right and then left until the car reaches an almost straight path for approximately 100 m. Then, the car turns right until the end of the street. The estimated length run of the first sequence is 166.07 m. In the second experiment the car starts turning left and then approaches a straight path for a while. After that, the car does a sharp right turn and moves straight during some meters, yielding an estimated length run of 216.33 m.

In figures 6(a) and 6(b) the two different trajectories and maps for the first sequence are displayed in a 2D view. In the same way, figures 7(a) and 7(b) depict the two different trajectories and maps for the second sequence.

The maps are composed of the 3D position of the features with its respective covariance, which has an elliptical shape. This covariance is an indicative of the quality of the map and the uncertainty in the estimate of the 3D position of the feature in the global map. The main result that can be obtained from Tab. I or just observing figures 6(b) and 7(b) is that the uncertainty in the 3D position of the features is much lower in the cases where an inverse depth parametrization is used. This is because as mentioned previously, the uncertainty of a far feature is much lower if it is parameterized as an inverse depth feature.

The estimated trajectory reflects well the exact shape of the real trajectory executed by the vehicle in both experiments. The trajectory for the first sequence is quite similar in both of the experiments. The estimated length is also similar in both experiments, the estimated length considering inverse depth parametrization is a little bit lower than the other case. However, in the second sequence the result considering an inverse depth parametrization reflects better the shape of the real trajectory and also the estimated length is closer to the ground truth. At the end of each experiment it can be observed that the quality of the trajectory is worst than at the beginning of the sequence, which is also reflected

in the final estimated length of the trajectory. This is because at the end of the sequences the number of landmarks in the EKF filter is so high (more than 300) that provokes inconsistency in the filter. This inconsistency in the EKF is due to the errors in the approximation of the observation model by a linearization, and also because the representation of the uncertanties and 3D feature position in a common global frame. Although this is not the purpose of this work, this problem can be solved by re-linearizing the filter after some error has been accumulated creating a new submap with a local coordinate frame, expressing the uncertainties and relative 3D positions according to this new local frame.

The main drawback of the inverse depth parametrization is the computational overload of representing a feature by 6 parameters instead of 3. This drawback can be important if real-time constraints are needed for the computation of each submap. Fig. 4 depicts the state vector size during some frames of the sequence 1, considering the two experiments. As it can be observed, the difference in size due to the overload of using an inverse depth parametrization is very significant as long as new features are added to the map.

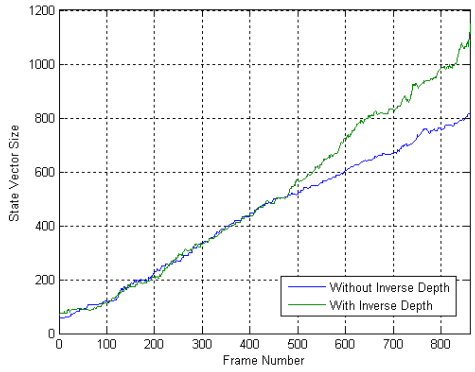


Fig. 4. Comparison of state vector size

VI. Conclusions and Future Works

In this paper we have presented a Visual SLAM approach that can estimate accurately the vehicle motion trajectory in urban roads considering small environments. In the same way a sparse map of high quality features is obtained. The system combines both bearing and depth information by means of two different types of feature parametrization: inverse depth and 3D. Inverse depth features can be switched efficiently to 3D features when its depth is below a depth threshold, reducing the uncertainty of the 3D position of far features in the global map, yielding a better localization.

We are very interested in studying the use of a dynamic threshold as a function of the kind of environment, instead of the static one that is currently used, so as to mantain the same map quality keeping real time constraints.

Considering 2D image templates and the normal vector of the plane that contains the point in the space improves the tracking considerably and it is better than using just 2D image templates. However, since the normal vector is only estimated once per feature (at the moment each feature is initialized), an update of the patch normals estimation would likely be of benefit.

In further works, a high level SLAM will be developed for mapping indoor and outdoor large environments fusing the infromation from our metric submaps. In addition, we are interested in fusing the stereo system with a commercial GPS for outdoor experiments in order to make the localization and mapping more robust, and compare our results with an accurate ground truth. In the same way, we will compare our Visual SLAM system with another techniques such as stereo Visual Odometry.

Acknowledgment

This work was supported in part by the Spanish Ministry of Education and Science (MEC) under grant TRA2005-08529-C02 (MOVICON Project) and grant PSE-370100-2007-2 (CABINTEC Project) as well as by the Community of Madrid under grant CM: S-0505/DPI/000176 (RoboCity2030 Project).

References

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2007.

[2] P. Mountney, D. Stoyanov, A. J. Davison, and G. Z. Yang, "Simultaneous stereoscope localization and soft-tissue mapping for minimally invasive surgery," *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2006.

[3] D. Schleicher, L. M. Bergasa, R. Barea, E. López, M. Ocaña, and J. Nuevo, "Real-time wide-angle stereo visual slam on large environments using sift features correction," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[4] D. Schleicher, L. M. Bergasa, R. Barea, E. López, and M. Ocaña, "Real-time simultaneous localization and mapping with a wide-angle stereo camera and adaptive patches," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[5] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.

[6] L. M. Paz, J. Guivant, J. D. Tardós, and J. Neira, "Data association in O(n) for divide and conquer SLAM," *Robotics: Science and Systems (RSS)*, 2007.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, 2007.

[8] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large scale 6DOF SLAM with stereo-in-hand," *IEEE Trans. Robotics*, vol. 24, no. 5, 2008.

[9] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE Trans. Robotics*, 2008.

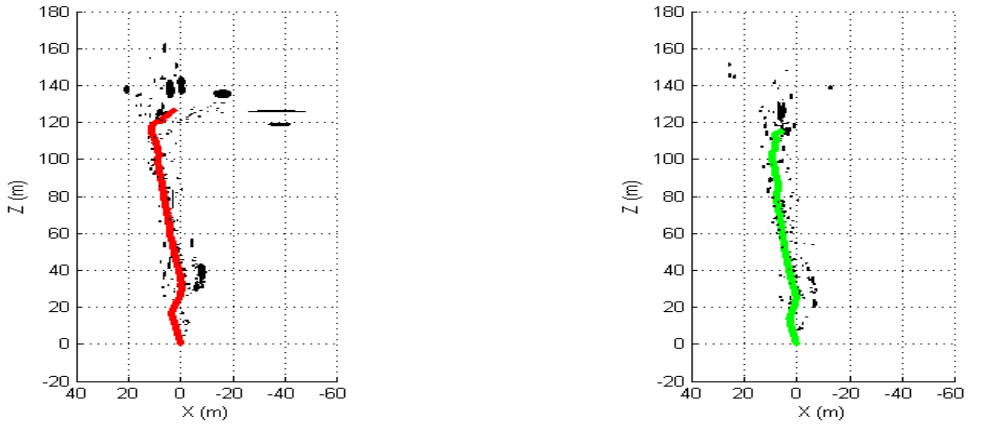
[10] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[11] B. Liang and N. Pears, "Visual navigation using planar homographies," *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2002.

[12] N. Molton, A. J. Davison, and I. Reid, "Locally planar patch features for real-time structure from motion," *British Machine Vision Conf. (BMVC)*, 2004.

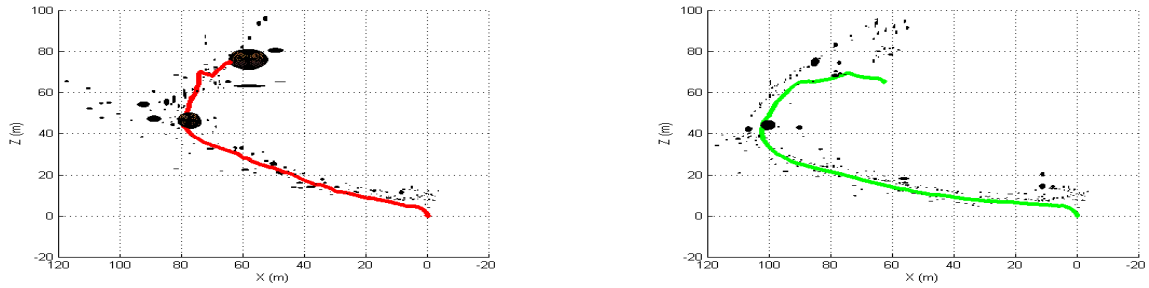
(a) Sequence 1 (b) Sequence 2

Fig. 5. Trajectories in the city for experiments 1 and 2



(a) Without Inverse Depth Par. (b) With Inverse Depth P. Z = 30 m

Fig. 6. Inverse Depth and 3D comparison: Sequence 1



(a) Without Inverse Depth Par. (b) With Inverse Depth P. Z = 30 m

Fig. 7. Inverse Depth and 3D comparison: Sequence 2

Seq.	**Case**	**% Inverse Features**	**Estimated Length (m)**	**Mean P_{YY} Trace**
1	Without Inverse Par.	0.00	133.97	2.4414
1	With Inverse Par., $Z_t = 30\ m$	12.25	129.08	0.7177
2	Without Inverse Par.	0.00	130.61	2.9729
2	With Inverse Par., $Z_t = 30\ m$	14.85	177.87	0.2188

TABLE I

Inverse Depth and 3D comparison: Estimated Length Run and Features Uncertainty

Session II

Multi-sensor perception & navigation

- **Title: Multi-Sensor Perception and Dynamic Path Planning in City Environments** *(invited paper)*
  Authors: Martin Rufli Luciano Spinello Roland Siegwart

- **Title: Camera and Laser Radar Co-detection of Pedestrians**
  Authors: Hao LI, Ming YANG, Huijia QIAN

- **Title: Model Based Vehicle Tracking in Urban Environments** *(invited paper)*
  Authors: Anna Petrovskaya and Sebastian Thrun

- **Title: Connexity based fronto-parallel plane detection for stereovision obstacle segmentation**
  Authors: Thomas Veit

- **Title: Safe and Dependable Operation of a Large Industrial Autonomous Forklift**
  Authors: Ashley Tews

Session II

Multi-sensor perception & navigation

- **Title: Multi-Sensor Perception and Dynamic Path Planning in City Environments** *(invited paper)*
  Authors: Martin Rufli Luciano Spinello Roland Siegwart

Multi-Sensor Perception and Dynamic Motion Planning in City Environments

Martin Rufli Luciano Spinello Roland Siegwart

Autonomous Systems Lab, ETH Zürich. Tannenstrasse 3, 8092 Zürich, Switzerland

martin.rufli@mavt.ethz.ch luciano.spinello@mavt.ethz.ch r.siegwart@ieee.org

***Abstract*— In this paper we describe a state lattice based motion planning approach, which we have successfully applied to large, cluttered, but *quasi-static* environments. Our approach produces smooth and complex maneuvers through the use of a multi-resolution state lattice, where the resolution is adapted based on the environment, and distance from the robot.**

We also describe a framework for detecting dynamic obstacles such as pedestrians and cars using a multisensor laser-camera detection and tracking method. Image detection is based on several extensions to the Implicit Shape Model technique; laser detection is instead achieved through the use of a Conditional Random Fields reasoning. Objects are tracked through the use of multiple motion model Kalman filters in order to cope with several different motion dynamics.

Urban environments, are complex, cluttered, and *dynamic* scenes, however. We therefore propose to extend our dynamic obstacle detection and tracking method with a short-term motion prediction functionality based on the same models used for tracking, effectively generating time based cost or risk maps. We further propose to implement these cost maps into our high-dimensional (5D to 6D) lattice planner to generate time-optimal trajectories in dynamic, cluttered environments. A D* implementation is envisioned to speed up re-planning dramatically.

I. INTRODUCTION

In urban environments, autonomous cars are required to navigate through both structured (streets) and unstructured (parking lots, off-road tracks) dynamic environments. Due to various vehicle and environmental constraints, they often need to plan complex maneuvers to perform this navigation. In order to provide responsive vehicle behavior in environments cluttered with dynamic obstacles, planning (and re-planning) needs to be performed in real-time (i.e. at 10Hz).

Until recently, these real-time objectives rendered the computation of optimal *and* kinematically feasible trajectories (i.e. to the next waypoint, several 100[m] away) infeasible. Instead, deterministic graph search algorithms such as Dijkstra [1] or A* [2] have often been applied to lower dimensional representations of the navigation problem (such as 2D grids). Paths returned by these grid based algorithms, although optimal on the grid, are not directly achievable due to vehicle constraints. Time consuming post-processing steps, such as path smoothing and the generation of a velocity profile along the smoothed path are required. Even if execution time is of no concern, in absence of (road) structure or in highly cluttered and dynamic environments, path- and velocity planning cannot be separated in order to arrive at truly time-optimal solutions, however.

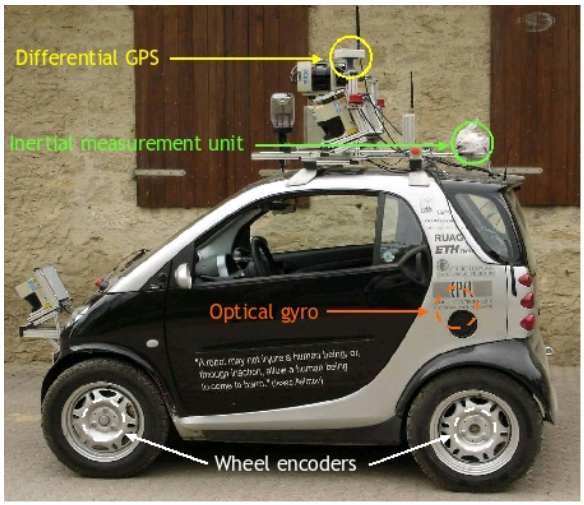


Fig. 1. SmartTer, the autonomous robotic car at the Autonomous Systems Lab, ETH Zurich.

A representation that addresses these issues comes in the form of the state lattice, a construct that reformulates the nonholonomic motion planning problem into graph search. Via the use of lattice segments (motion primitives), the enormous search space is reduced dramatically in an intelligent way so as to comply with vehicle kinematic constraints and retain all feasible vehicle motions (down to a discretization limit) [3].

Lattices have been successfully applied in various contexts of *path* planning, such as low-velocity rough-terrain navigation [3], and high-velocity navigation in urban environments [4], [5] and [6]. Recently, Kushleyev *et al.* [7] introduced the concept of the time-bounded lattice, which adds vehicle velocity and time as separate search dimensions to the multi-resolution lattice state space, thereby merging short-term planning in time with long-term planning without time.

When incorporating time-based predictions into planning, solutions generated can be vastly superior to non time-parametrized paths, but also unsafe (a process running in parallel could be implemented to ensure safety, if desired). The fidelity of the predictions therefore plays a key role in solution quality.

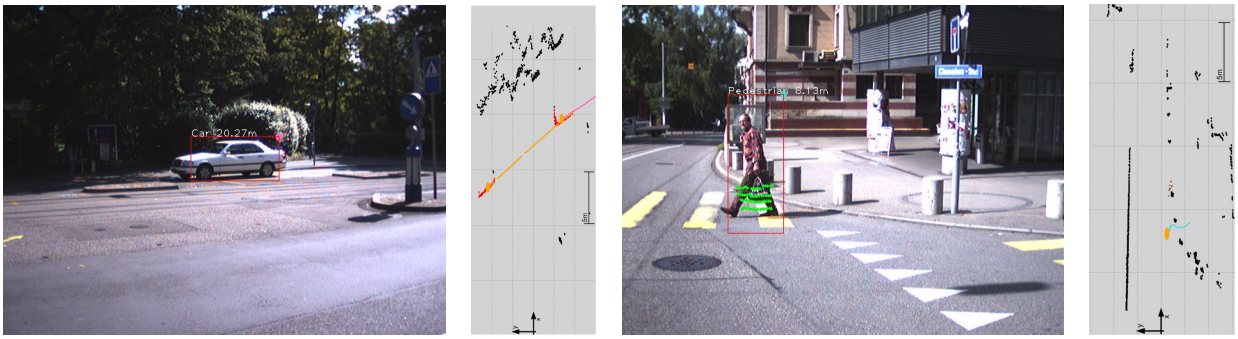


Fig. 2. A typical urban environment detection and tracking example: cars and pedestrian are correctly detected. Tracks are shown in the gray plot.

Urban environments feature many different types of dynamic obstacles, most importantly pedestrians and cars (which exhibit very dissimilar dynamics). Pedestrians are particularly difficult to detect because of their high variability in appearance due to clothing, illumination and self occlusions. Cars are wide objects that dramatically change their visual characteristics when the viewpoint changes. It is thus advantageous to detect (label) and track these objects, in order to generate a more informed prediction of their future movement.

In the detection part we employ a multisensor laser-camera detection and tracking method. Image detection is based on several extensions to the Implicit Shape Model technique [8]; laser detection is instead achieved through the use of a Conditional Random Fields reasoning. Objects are then tracked through the use of multiple motion model Kalman filters in order to cope with several different motion dynamics. These motion models could then further be used to predict the near-future behavior of dynamic objects by effectively generating time based cost or risk maps. We propose to implement such time based cost maps into our high-dimensional (5D to 6D) lattice planner and generate time-optimal trajectories in dynamic, cluttered environments.

Several approaches can be found in the literature to identify a person in 2D laser data [9], [10], [11], [12] and [13]. In the area of image-based people detection, there mainly exist two kinds of approaches, one uses the analysis of a *detection window* or *templates* [14], [15], the other performs a *parts-based* detection [16], [17], [18]. Existing people detection methods based on camera *and* laser rangefinder data either use hard constrained approaches or hand tuned thresholding [19], [20]. The only work present in literature that combines images and laser by using Conditional Random Fields is the work of Douillard [21] that uses image features in order to enhance object detection but it does not explicitly handle occlusions and separate image detection hypotheses.

In Section II, we describe our multi-sensor detection and tracking method. Section III introduces our multi-resolution state lattice. We also propose further steps on how to combine these two frameworks in order to arrive at real-time planning capability in cluttered and dynamic city environments. Section IV concludes this paper and proposes further work.

II. DETECTION OF SENSITIVE URBAN OBJECTS: PEDESTRIANS AND CARS

A. Overview of the method

Our system is composed of three main components: an appearance based detector that uses the information from camera images, a 2D-laser based detector providing structural information, and a tracking module that uses the combined information from both sensor modalities and provides an estimate of the motion vector for each tracked object. The laser based detection applies a Conditional Random Field (CRF) on a boosted set of geometrical and statistical features of 2D scan points. The image based detection system extends a multiclass version of the Implicit Shape Model (ISM) [22] and uses Shape Context descriptors [23] computed at Harris-Laplace and Hessian interest points. It also uses the laser based detection result projected into the image to constrain the position and scale of the detected objects. Then, the tracking module applies an Extended Kalman Filter (EKF) to the combined detection results where two different motion models are implemented to account for a high variety of possible object motions. In the following, we describe the particular components.

B. Appearance Based Detection

Our image-based people detector is mostly inspired by the work of Leibe *et al.* [18] on scale-invariant Implicit Shape Models (ISM). Shortly, an ISM consists in a set of local region descriptors, called the *codebook*, and a set of displacements and scale factors, usually named *votes*, for each descriptor. The idea of the votes is that each descriptor can be found at different positions inside an object and at different scales, and thus a vote points from the position of the descriptor to the center of the object as it was found in the training data set. To obtain an ISM from labeled training data, all descriptors are first clustered, usually using agglomerative clustering, and then the votes are computed by adding the scale and the displacement of the objects' center to the descriptors in the codebook. For the detection, new descriptors are computed on a given test image and matched against the descriptors in the codebook. The votes that are cast by each matched descriptor are collected in a 3D *voting space*, and a maximum density estimator is used to find the most likely position and scale of an object.

1) ISM Extensions: We introduce the following novelties in the image detection part. Extensions in the Learning Phase:

- **Learning of Subparts:** The aim of this procedure is to enrich the information that is obtained from the voters by distinguishing between different object subparts from which the vote was cast.
- **Learning a Template Mask:** The idea here is to build a probabilistic template map from the individual segmentation masks in the training set in order to reject early object hypotheses.
- **Learning Superfeatures:** We here propose a method to drive the detection while still maintaining information richness. The idea is to find good features in the image space (namely $\langle x, y, scale \rangle$) and descriptor space ($n$-d space) that could vote for the object center with more weight to ease the hypothesis selection.

Extensions in the Testing Phase:

- **Using Superfeatures:** *Superfeatures* and features vote for object centers in the same voting space: the votes generated by the first are bigger than the latter.
- **Using subparts and prob. template in the cost function:** Each hypothesis is now defined by an angular histogram in which the bins are defined by the subparts. Moreover, the probabilistic template is used to prune feature matches that lie far outside the probabilistic shape (scaled according to the hypothesis). We employ a maximum likelihood estimation method in order to select the winning hypotheses.
- **Discriminate between object classes:** We employ a common measure to do hypothesis selection by using the probabilistic template area ratio: each assigned feature for a certain hypothesis occupies a scaled square area in the probabilistic template.

C. Structure Based Detection

We modeled the object detection problem in laser data as a Conditional Random Field (CRFs) probability inference [24]. CRFs represent the conditional probability $p(\mathbf{y} \mid \mathbf{z})$ using an undirected cyclic graph, in which each node is associated with a hidden random variable y_i and an observation $\mathbf{z}_i$. In our case, the y_i is a discrete label that ranges over 2 different classes (pedestrian and car) and the observations $\mathbf{z}_i$ are 2D points in the laser scan.

For the likelihood minimization in the training phase we use the L-BFGS gradient descent method [25]. Once the weights are obtained, they are used in the inference phase to find the labels $\mathbf{y}$ that maximize:

$$p(\mathbf{y} \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{i=1}^{N} \varphi(\mathbf{z}_i, y_i) \prod_{(i,j)\in\mathcal{E}} \psi(\mathbf{z}_i, \mathbf{z}_j, y_i, y_j), \quad (1)$$

Here, we do not need to compute the partition function Z, as it is not dependent on $\mathbf{y}$. We use max-product loopy belief propagation to find the distributions of each label y_i. The final labels are then obtained as those that are most likely for each node.

1) Node, Edge Features and Connectivity: As node features $\mathbf{f}_n$ we use a set of statistical and geometrical features such as height, width, circularity, standard deviation, kurtosis, etc. (for a full list see [26]). We compute these features in a local neighborhood around each point, which we determine by jump distance clustering. We can then use these features as an input to the CRF classification algorithm. However as stated in [27], and also from our own observation, the CRF is not able to handle non-linear relations between the observations and the labels, which is a consequence of the log-linear model described above. To overcome this problem, we apply AdaBoost [28] to the node features and use the outcome of AdaBoost as features for the CRF. Nodes are connected using a Delaunay triangulation.

2) Tracking objects for sensor fusion: In order to fuse the information coming from both sensors (camera and laser) and to simultaneously keep track of the object we use an EKF based tracking system, first introduced in [29]. Here, each object is tracked with several motion models (in this case: Brownian motion and linear velocity) in order to cope with pedestrian and car movements. We perform tracking in the laser data, therefore camera detections are projected and assigned to segments in the laser data. In order to reliably track wide objects, like cars, tracking single segments are not enough. Single segments tend to be spatially very unstable due to the noise present in outdoor environments and the scatter resulting from the distance with respect to the observer. We therefore group segments with the same class label using Delaunay triangulation and a trim distance rule. The resulting cluster will have a more stable position and a probability of being a class that is the average of its members. Each Kalman filter state ($\langle x, y, (v_x, v_y) \rangle$) is augmented with N states where N is the number of classes present in the detector. Indeed, the observation vector z fed to the tracking system consists of the position of the cluster and the class label probability. The matrix H that models the observations to mapping in the Kalman Filter $x = Hz$ is defined by $H = [H_{lsr}; H_{cam}]$ in order to manage multiple inputs from different sensors. Each track, with its relative prediction, can be used for the planning navigation algorithm.

Quantitative fused detection and tracking results are shown in Fig 4 in which a Recall/false positive per frame is depicted. The test dataset is composed in total of 511 frames. The car detection obtains superior results with respect to pedestrian detection specially due to the non flexible shape which guarantees more distinct descriptors in the codebook. Both graphs clearly show the advantage of using a combined laser-camera detection method with respect to single modal techniques.

III. SEARCH IN DYNAMIC ENVIRONMENTS

A. Overview

Deterministic motion planning in dynamic environments remains extremely challenging due to the increased dimensionality in state space. Lattices allow for an intelligent reduction in search space, thereby increasing solution speed dramatically, without noticeably affecting solution quality.

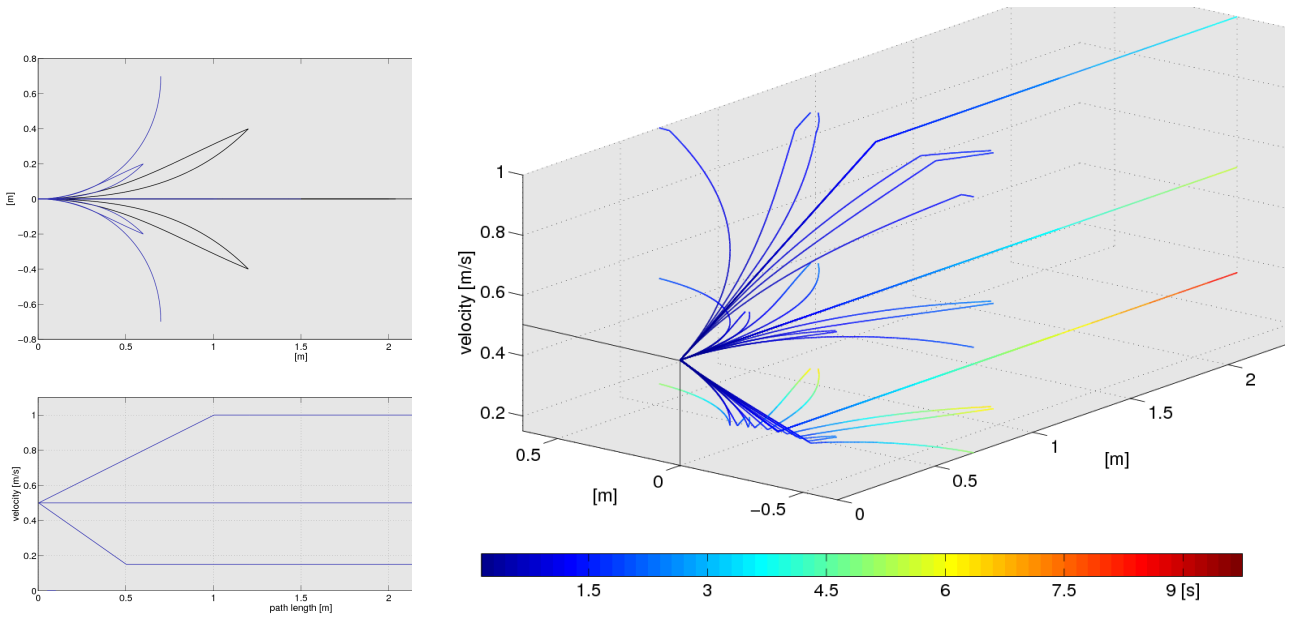


Fig. 3. Example of a 16-directional multi-resolution lattice (one of the 16 initial headings shown), which was constructed for a diff-drive household robot. We plan on generating a similar lattice for our autonomous car, albeit with velocities of up to 20[m/s], and the steering angle considered as an additional dimension. Top Left: black segments denote the low-resolution part. Black and blue segments together denote the high-resolution part. Some of the short straight segments are occluded. Bottom Left: velocity profiles as a function of path length. Right: full 4D (x, y, θ, v) view for one initial heading starting at a velocity of 0.5[m/s].

They are thus well suited for deterministic high-dimensional planning tasks.

Here, we describe a generic lattice generation process, and detail our extension to a multi-resolution lattice, which allows for further reduction in search space. We then describe the search over the obtained lattice and how dynamic short-term predictions of dynamic obstacles' future movement can be implemented into it. Finally, we sketch an approach for speeding up re-planning times, once an initial plan is available.

B. Lattice Generation

The base lattice is generated by forward simulating a suitable vehicle model using a (uniform) sampling of desired velocity and steering angle as model inputs to generate a large set of motion primitives, followed by a careful selection of a small subset of these primitives: the selection is performed in such a way, that through combination of motion primitives belonging to the subset, any and all original motions can be reconstructed down to a discretization limit (see [3] for an automated approach). Compliance with vehicle kinematic and dynamic constraints is inherently guaranteed, given the above design iteration and a high-fidelity dynamic model of the vehicle.

The state lattice is then constructed by starting with a node in a desired configuration space (in the case of a car: $(x, y, heading, v, steeringangle)$) and then creating edges emanating from this node. From every node these edges transition to, the process is repeated, resulting in a connected graph consisting of all the nodes and edges generated. Typically, a discretization is applied to the nodes so that they all reside on some grid in the configuration space. By choosing a variable discretization, search speed is further increased.

Fig. 3 shows our currently implemented indoor differential-drive lattice, operating at 16 directions on a 0.1[m] grid and 4 discrete velocities up to 1[m/s]. In the future, we plan on implementing a similar lattice for Ackermann-like vehicles and increase the velocity discretization substantially.

1) Multi-Resolution Lattice Extension [Partially Implemented]: An important factor when designing a state lattice is the resolution of the discretization used to represent the nodes in the lattice in terms of the *position, heading, and velocity discretization*. From a *computational* point of view, the underlying position discretization should be chosen as coarse as possible. From a *completeness* point of view, it needs to be fine enough to generate feasible plans in narrow areas. 0.25[m] position discretization allows for successful parking maneuvers at low velocities [4]. State lattices have been successfully employed with 8 to 64 (uniform) directions. We found that for smooth *and* natural looking paths, generally 32 directions are required.

Expanding 32-directional lattice segments over a 0.25[m] grid is computationally intensive, however, and in many cases not necessary to achieve a smooth and feasible trajectory. A recently proposed solution to this problem is to incorporate a *multi-resolution* lattice, operating at two or more resolutions based on task [4], and environmental characteristics [30]. The lower resolution is usually chosen as a subset of the higher resolution lattice so that suboptimality guarantees can be given with respect to the low-resolution lattice. In our

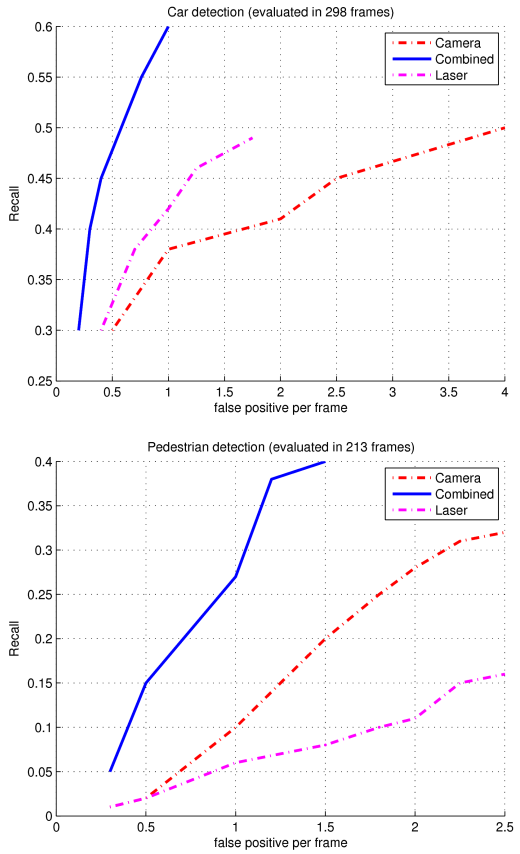


Fig. 4. Recall/false positive per frame graph for cars and pedestrian in a test dataset.

forthcoming lattice we plan on implementing the following ideas:

1) Variable heading, position, and velocity discretization, based on distance from the robot position. In combination with frequent replanning, the robot is guaranteed to always remain on a high-fidelity trajectory, while the trajectory assumes a lower quality further away, although it remains feasible.
2) Variable position discretization based on vehicle velocity. At higher velocities, safety margins around the vehicle should be increased, hence a coarser discretization is justified.
3) Addition of time into the state space, in conjunction with short-term prediction of dynamic obstacles' motion. By incorporating the ideas recently presented by Kushleyev *et al.* [7], the time dimension can be dropped at a variable time horizon. Unlike in their approach, we do not believe in dropping all but the position dimensions at that instant, as the trajectory would loose its feasibility property.

These considerations are expected to have the effect of generally producing trajectories close to the optimal one (had only the high-resolution lattice been expanded) and at the same time generating solutions much faster than by expanding the high-resolution lattice alone [4].

C. Search over the Lattice

A state lattice can be considered a specific method for constructing a directed graph. Thus, regular graph search algorithms are applicable for searches along a state lattice. The most popular and efficient deterministic graph search algorithms belong to the family of 'best-first' searches. They expand promising states first, making use of a heuristic function to guide them, the original one being A* [2]. Subsequently, more efficient algorithms (particularly for replanning) have been devised (see i.e. [31] for an overview). We currently use a forward planning Anytime A* extension [32] over the presented multi-resolution state lattice, although there are plans on implementing a backward planning algorithm for more efficient replanning (see Section III-C.5).

1) Heuristic generation: The more accurate a heuristic is, the faster the following high-dimensional search will converge to the optimal solution. To this end, we perform a 2D Dijkstra search out from the goal location(s) so that the cost values become the heuristic values for the high-dimensional search. To ensure the admissibility of this heuristic, we constrain the costs of actions to be upperbound by their Euclidean distance. In particular, the obtained heuristic costs are divided by 1.03 for the 16-neighborhood employed (see [33] for details).

2) Planning with quasi-static obstacles: Let us define quasi-static obstacles as obstacles with a low maximal velocity compared to our vehicle's dynamics. In such cases, combined with frequent replanning, the obstacles' current (inflated) pose can be considered untraversable during both the heuristic generation and the higher-dimensional planning step. Quantitative results show, that in such cases, our 4D (x, y, θ, v) lattice planner is able to generate local, smooth and feasible *trajectories* in real-time; our 3D (x, y, θ) planner is able to generate (global,) smooth, and feasible *paths* in real-time (see Fig. 5, fig. 6 and [33] for more details).

3) Representation of dynamic obstacles [Planned]: Representing dynamic obstacles during the planning stage remains a challenging problem. For short-term predictions of dynamic obstacles' future trajectories, the same motion models could be used as for tracking (see Section II). In the case of pedestrians and vehicles, we hope to provide meaningful predictions several seconds into the future. We then plan on mapping these predictions into local time-parametrized cost or risk maps: through a discretization in time (i.e. 0.1[s]) a set of local maps is populated, where a given space-time entry is assigned the sum of all detected dynamic obstacles' probabilities to occupy this location at the given time (cost map), multiplied with a severity factor corresponding to the associated objects' types (risk map).

Further work in this direction may involve quantifying prediction fidelity, extending the prediction horizon, and modelling agent-agent interactions.

4) Planning with dynamic obstacles [Planned]: Planning with dynamic obstacles by considering their predicted future motion amounts to applying the same search algorithm as in the static situation (Anytime A* in our case), albeit in a higher-dimensional space with time added as a separate

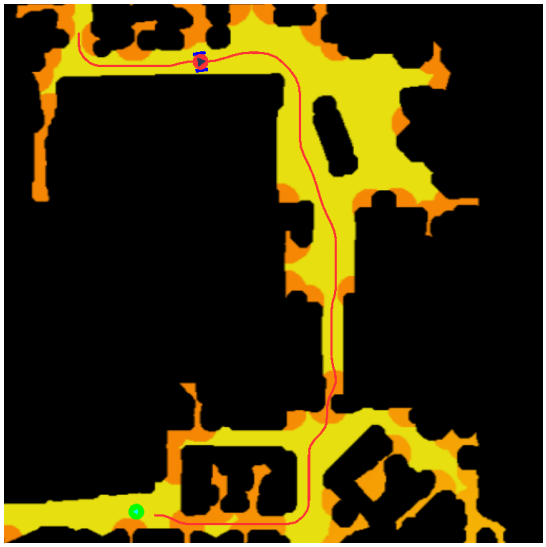


Fig. 5. The 'long run' encompasses a path of approximately 40[m] through the Intel Research Pittsburgh lab. The 3D planner expands less than 100 states in 0.04[s] to arrive at a solution. The 4D planner takes 2.28[s] with 5900 expanded states (pictured). The large difference in performance lies in the 2D heuristic, which does not well predict the influence, which areas of reduced maximal velocity (i.e. in vicinity of doors and narrow hallways, colored orange) have on the 4D planner.

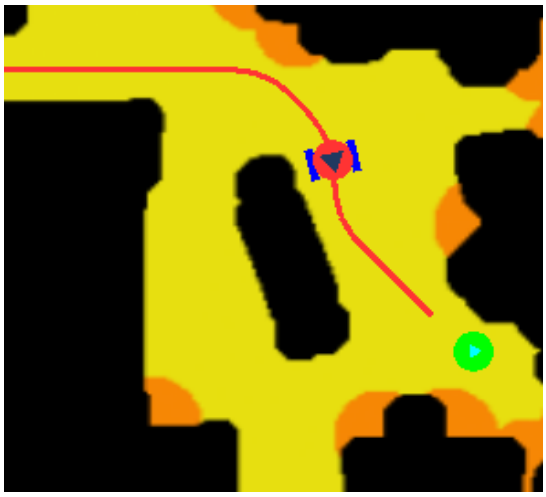


Fig. 6. The 'short run' encompasses a path in the Intel Research Pittsburgh kitchen area (approx. 10[m]). Both the 3D (less than 100 expansions in 0.01[s]) and the 4D planner (less than 100 expansions in 0.03[s], pictured) show real-time capability in this scenario.

search dimension. Note that due to the probabilistic nature of obstacles' predicted motion, a generated trajectory is not guaranteed collision free. In many situations it might be useful to have such guarantees however. In such cases, a collision avoidance algorithm could be run on the controller level.

5) Speeding-up replanning [Planned]: Best-first searches, starting their expansions at the goal location (i.e. D* [31]), have been shown to be one to two orders of magnitude faster than forward planning algorithms when replanning in only slightly updated environments. Perceived changes in the environment often appear close to the robot (due to proprioceptive sensors). By planning from a distant goal, large parts of the previously generated solution thus remain valid. Despite these advantages, we currently do not use backward searching algorithms:

1) Due to uncertainty in dynamic obstacles' predicted motion, large portions of the local map need to be updated during every timestep, rendering repairing algorithms ineffective.
2) When expanding states from the goal, the exact time of arrival can only be induced once a solution is found. During a given time of the search, however, it is unclear which of the local time-parametrized cost/risk maps needs to be called.

On the other hand, local high-dimensional plans are typically generated over up to several hundred meters and dramatic changes in costmaps are only perceivable in close vicinity of the robot, where time-based planning is performed. Further away, only slight changes are expected due to newly perceived or updated static obstacles. Additionally, we found a promising solution to the time-association problem by producing an enormous look-up table, which encodes all feasible motions up to a certain time horizon. It grows exponentially with the branching factor of the lattice, however, currently limiting it to a horizon below 5[s]. Further tests will show, whether such a look-up table based backward planning algorithm is able to outperform our current ARA* implementation.

IV. CONCLUSIONS AND FURTHER WORK

A. Conclusions

In this paper we presented a multi-resolution state lattice based path planning approach, which produces smooth maneuvers in large and complex environments.

We also presented a framework for detecting dynamic obstacles such as pedestrians and cars using a multisensor camera-laser detection method based on several extensions to the Implicit Shape Model technique, and Conditional Random Fields reasoning. Tracking was demonstrated through the use of multiple motion model Kalman filters.

Both of these methods have been extensively tested and work well in isolation.

B. Further Work

Further work thus mainly lies in combining and extending the two presented methods:

1) Tracked pedestrians' and cars' future motions need to be estimated, possibly using the same motion models used for tracking.
2) A multi-resolution state lattice for Ackermann-like vehicles needs to be designed and implemented into our SmartTer platform.
3) A look-up table based backward planning algorithm, which potentially decreases search time substantially, may be considered.

REFERENCES

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation (Vol. 1)*, pages 375–382. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[3] M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2005.

[4] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2008.

[5] D. Ferguson, T. Howard, and M. Likhachev. Motion Planning in Urban Environments: Part I. In *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, September 2008.

[6] T. Howard D. Ferguson and M. Likhachev. Motion Planning in Urban Environments: Part II. In *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, September 2008.

[7] A. Kushleyev and M. Likhachev. Time-bounded lattice for efcient planning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[8] L Spinello, R Triebel, and R Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, 2008.

[9] D. Schulz, W. Burgard, D. Fox, and A. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *Int. Journ. of Robotics Research (IJRR)*, 22(2):99–116, 2003.

[10] E. A. Topp and H. I. Christensen. Tracking for following and passing persons. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, 2005.

[11] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, pages 3930–3935, 2005.

[12] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2003.

[13] Kai O. Arras, Óscar Martínez Mozos, and Wolfram Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2007.

[14] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.

[15] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE Int. Conf. on Computer Vision (ICCV)*, page 734, Washington, DC, USA, 2003. IEEE Computer Society.

[16] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 66–73, 2000.

[17] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. Journ. of Comp. Vis.*, 43(1):45–68, 2001.

[18] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 878–885, Washington, DC, USA, 2005. IEEE Computer Society.

[19] Z. Zivkovic and B. Kröse. Part based people detection using 2d range data and images. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, San Diego, USA, November 2007.

[20] D. Schulz. A probabilistic exemplar approach to combine laser and vision for person tracking. In *Robotics: Science and Systems (RSS)*, Philadelphia, USA, August 2006.

[21] Bertrand Douillard, Dieter Fox, and Fabio Ramos. Laser and vision based outdoor object mapping. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.

[22] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2007.

[23] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. volume 24, pages 509–522, 2002.

[24] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *Proc. of the INt. Conf. on Machine Learning (ICML)*, 2001.

[25] D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)), 1989.

[26] L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2008.

[27] Fabio Ramos, Dieter Fox, and Hugh Durrant-Whyte. Crf-matching: Conditional random fields for feature-based scan matching. In *RSS*, 2007.

[28] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[29] L Spinello, R Triebel, and R Siegwart. Multimodal people detection and tracking in crowded scenes. In *Proc. of The AAAI Conference on Artificial Intelligence*, July 2008.

[30] M. Rufli, D. Ferguson, and R. Siegwart. Smooth path planning in constrained environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[31] D. Ferguson, M. Likhachev, and A. Stentz. A guide to heuristic-based path planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.

[32] M. Likhachev, G. Gordon, and S. Thrun. Ara*: Anytime a* with provable bound son sub-optimality. In *Advances in Neural Information Processing Systems, MIT Press*, 2003.

[33] M. Rufli. Smooth path planning in dynamic environments. In *Master Thesis, ETH Zurich*, 2008.

Session II

Multi-sensor perception & navigation

- **Title: Camera and Laser Radar Co-detection of Pedestrians**
  Authors: Hao LI, Ming YANG, Huijia QIAN

Camera and Laser Scanner Co-detection of Pedestrians

Hao LI, Ming YANG, Huijia QIAN

ABSTRACT—Intelligent vehicle technology is a promising technology for enhancing urban traffic safety and efficiency. Pedestrian detection is an important issue for applications of intelligent vehicles in urban environments. The kind of most widely used method for pedestrian detection is vision based method. One general problem for vision based method is how to efficiently locate a proper ROI (region of interests) which contains a candidate object. Another problem is how to detect and segment features of candidate objects out of ROI. In this paper, a camera and laser scanner co-detection method is proposed. First, a method of camera and laser scanner co-calibration is presented. Second, a method of how to obtain proper ROI and the contours of candidate objects using the co-calibration results is introduced. Finally, a decision rule is induced from a set of examples of contour shapes of both pedestrians and landmarks (They are most likely to be confused with each other because of their similarity in size). Some experimental results are given for validating the camera and laser scanner co-detection method.

1 Introduction

Intelligent vehicle technology is a promising technology for enhancing urban traffic safety and efficiency. Since there are lots of pedestrians in urban environments, the problem of how to ensure the safety of pedestrians arises urgently in the application of intelligent vehicles in urban environments. To ensure the safety of pedestrians, the intelligent vehicle system should detect pedestrians nearby correctly and in time. Therefore, pedestrian detection is an important issue for applications of intelligent vehicles in urban environments.

The problem of pedestrian detection is a considerable challenge. For this problem, the kind of most widely used method is vision based method. Broggi et al [1] proposes a method mainly based on human shape features, especially the vertical edge symmetry and binary model, which are used to localize pedestrians' heads through stereo-vision's distance refinement. Gavrila [2] uses a method of hierarchical template matching (a lot of pedestrians' templates are needed) based on contour figures and intensity features. Curio et al [3] presents a hybrid method architecture which integrates texture information, entropy, template matching results and etc. Shashua et al [4] presents a two-steps detection method in which the Adaboost training is used for the single-frame detection step and a final decision is made by integrating information of multi-frames. Besides vision based method, laser scanner based method has also been reported [5].

The process of pedestrian detection can be mainly divided into two steps: object localization and object classification. In the first step, candidate objects are localized (obtain ROI) and segmented out from sensor data space (segment contours of candidate objects out of ROI). In the second step, a decision on whether a candidate object is a pedestrian is made based on the decision rule. The general difficulties for vision based method are at the step of object localization, while the general difficulties for laser scanner based method are at the step of object classification. Therefore, a more robust method might be realized if both camera and laser scanner are used together and cooperate with each other.

In this paper, a new camera and laser scanner co-detection method is proposed. It mainly consists of three parts: 1) a method of camera and laser scanner co-calibration; 2) a method of how to obtain proper ROI and the contours of candidate objects using the co-calibration results; 3) a decision rule induced from a set of examples of contour shapes of both pedestrians and landmarks. The paper is organized as follows: a method of camera and laser scanner co-calibration is presented in section2; the co-detection method is proposed in section3; experimental results on the co-calibration method and co-detection method are given in section4, followed by a conclusion in section5.

2 Co-calibration of camera and laser radar

The co-calibration is to determine the geometric transform relationship between three coordinates, i.e. the laser scanner coordinate, the image coordinate and the vehicle coordinate.

2.1 The three coordinate systems

The image coordinate is a 2D rectangular coordinate, denoted by a pair (u, v), where u, v mean the rows, columns of a pixel point.

The laser scanner used is a 2D laser scanner. The laser scanner coordinate mentioned here is a 3D rectangular coordinate, denoted by a triplet (x_p, y_p, z_p), where the origin point O_p is at the emitting point of the laser scanner; the X_p-axis and the Y_p-axis are on the scanning plane of the laser

Hao LI is with Department of Automation, Shanghai Jiao Tong University, No.800 Dong Chuan Road, Shanghai, 200240, P.R. China (haoli@sjtu.edu.cn)

Ming YANG is with Department of Automation, Shanghai Jiao Tong University, No.800 Dong Chuan Road, Shanghai, 200240, P.R. China (MingYANG@sjtu.edu.cn)

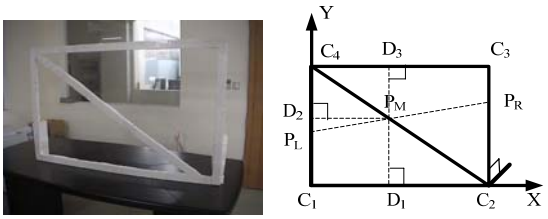
Huijia QIAN is with Department of Automation, Shanghai Jiao Tong University, No.800 Dong Chuan Road, Shanghai, 200240, P.R. China (qianhuijia23@yahoo.com.cn)

scanner; the Z_p-axis satisfies right-hand rule with the X_p-axis and the Y_p-axis.

The vehicle coordinate is a 3D rectangular coordinate, denoted by a triplet (x_w, y_w, z_w), where the X_w-axis and the Y_w-axis are on the ground surface; the origin point O_w is right under the front center of the vehicle; the X_w-axis is in the longitudinal direction of the vehicle while the Y_w-axis is in the lateral direction; the Z_w-axis satisfies right-hand rule with the X_w-axis and the Y_w-axis.

2.2 The calibration tool and the method of obtaining control points

Some control points are needed for the co-calibration. But the laser beam is invisible, so control points can not be determined directly and they can only be determined indirectly using certain calibration tool. The calibration tool designed is shown in Fig1(a). It is a rectangular frame with one diagonal connected. A bracket on the frame bottom corner is used to hold the rectangular frame perpendicular to the ground surface where it is put.



(a) (b)

Fig 1 the calibration tool and control points

Suppose the scanning plane of the laser scanner intersects the rectangular frame at point P_L, P_M and P_R; the line $D_3P_MD_1$ is perpendicular to C_1C_2; the line P_MD_2 is perpendicular to C_1C_4; as shown in Fig1(b). Although the intersected point P_M is invisible, its place can be revealed indirectly by geometric knowledge:

$$P_M D_2 = C_1 D_1 = \frac{P_L P_M}{P_L P_M + P_M P_R} C_1 C_2$$

$$P_M D_1 = C_1 D_2 = \frac{P_M P_R}{P_L P_M + P_M P_R} C_1 C_4$$

Where the lengths of C_1C_2 and C_1C_4 are measured directly and the lengths of P_LP_M and P_MP_R are computed from the coordinate information of P_L, P_M, P_R in the laser scanner coordinate.

If a 2D rectangular coordinate (called frame coordinate) is established on the rectangular frame of the calibration tool, as shown in Fig1(b), then the coordinates of C_1, C_2, C_3, C_4 in the frame coordinate are respectively (0,0), (C_1C_2,0), (C_1C_2,C_1C_4), (0,C_1C_4). The image coordinates of C_1, C_2, C_3, C_4 are obtained manually. The geometric transform relationship between the frame coordinate and the image coordinate can be described by a homography transform which can be computed with the coordinate information of C_1, C_2, C_3, C_4 [6].

The coordinates of D_1, P_M, D_3 in the frame coordinate are respectively (C_1D_1,0), (C_1D_1,C_1D_2), (C_1D_1, C_1C_4). The image coordinates of D_1, P_M, D_3 can be computed through the homography transform.

The Z_w coordinate of D_1, P_M, D_3 in the vehicle coordinate are respectively 0, C_1D_2, and C_1C_4. Their X_w, Y_w coordinates in the vehicle coordinate are the same and are measured directly.

In sum, at each place where the calibration tool is put, the coordinates of P_M in the laser scanner coordinate, the image coordinate and the vehicle coordinate are all known; the coordinates of D_1, D_3 in the image coordinate and the vehicle coordinate are known. Therefore, P_M can be used as control point for calibrating the geometric transform relationship between the laser scanner coordinate and the vehicle coordinate, while P_M, D_1, D_3 can be used as control point for calibrating the geometric transform relationship between the vehicle coordinate and the image coordinate.

2.3 Computing the geometric transform relationship between the three coordinate systems

(a) The geometric transform relationship between the laser radar coordinate and the vehicle coordinate

Suppose there are N control points and their coordinates in the laser scanner coordinate and the vehicle coordinate are respectively (x_{pi}, y_{pi}, z_{pi}) and (x_{wi}, y_{wi}, z_{wi}); i=1, 2, … , N. The geometric transform between the laser scanner coordinate and the vehicle coordinate can be described by a rotation and translation transform:

$$\begin{bmatrix} \mathbf{R_{pw}} & \mathbf{T_{pw}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{pi} \\ y_{pi} \\ z_{pi} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{wi} \\ y_{wi} \\ z_{wi} \\ 1 \end{bmatrix} \quad (1)$$

Where $\mathbf{R_{pw}}$ is the rotation matrix and $\mathbf{T_{pw}} = [T_x, T_y, T_z]^T$ is the translation vector. Any rotation matrix can be expressed in terms of 3 independent parameters (R_x, R_y, R_z), i.e. $\mathbf{R_{pw}}$=exp($\mathbf{W_R}$).

$$\mathbf{W_R} = \begin{bmatrix} 0 & -R_z & R_y \\ R_z & 0 & -R_x \\ -R_y & R_x & 0 \end{bmatrix}$$

Because of the nonlinearity of $\mathbf{R_{pw}}$ with respect to R_x, R_y, R_z, it is not easy to compute R_x, R_y, R_z and T_x, T_y, T_z directly. An iteration method is needed to compute R_x, R_y, R_z along with T_x, T_y, T_z. Suppose the result of $\mathbf{W_R}$ after k rounds of iteration is $\mathbf{W_R}(k)$. The initial value $\mathbf{W_R}(0)$ can be estimated quantitatively. The basic idea is: regard $\mathbf{R_{pw}}$ as a matrix of 9 independent parameters; then a linear equation group of these 9 parameters and T_x, T_y, T_z can be derived using Eq.(1) and the coordinate information of control points; $\mathbf{R_{pw}}$ is obtained by solving this linear equation group; then $\mathbf{W_R}(0)$ can be estimated as ($\exp^{-1}(\mathbf{R_{pw}})$-$\exp^{-1}(\mathbf{R_{pw}})^T$)/2. Details are not introduced here. In fact, since the requirement of the accuracy of $\mathbf{W_R}(0)$ is not high, $\mathbf{W_R}(0)$ can be just estimated qualitatively according to the installation position of the laser scanner.

The method of refining $\mathbf{W_R}$ with iterations is introduced as follows. Denote $R_x(k+1) = R_x(k)+\Delta R_x(k)$; $R_y(k+1) = R_y(k)+\Delta R_y(k)$; $R_z(k+1) = R_z(k)+\Delta R_z(k)$; and $\mathbf{W_R}(k+1) = \mathbf{W_R}(k)+\Delta\mathbf{W_R}(k)$. Let $\exp(\mathbf{W_R}(k)+\Delta\mathbf{W_R}(k)) \approx$

$(\mathbf{I}+\Delta\mathbf{W_R}(k))\exp(\mathbf{W_R}(k))$, and substitute it into Eq.(1)

$$\begin{bmatrix} 1 & -\Delta R_z(k) & \Delta R_y(k) \\ \Delta R_z(k) & 1 & -\Delta R_x(k) \\ -\Delta R_y(k) & \Delta R_x(k) & 1 \end{bmatrix}\begin{bmatrix} x_{pi}(k) \\ y_{pi}(k) \\ z_{pi}(k) \end{bmatrix}+\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}=\begin{bmatrix} x_{wi} \\ y_{wi} \\ z_{wi} \end{bmatrix} \quad (2)$$

Where $[x_{pi}(k),y_{pi}(k),z_{pi}(k)]^{\mathrm{T}} = \exp(\mathbf{W_R}(k))\ [x_{pi},y_{pi},z_{pi}]^{\mathrm{T}}$. Then from Eq.(2) it can be derived:

$$\begin{bmatrix} 0 & z_{pi}(k) & -y_{pi}(k) & 1 & 0 & 0 \\ -z_{pi}(k) & 0 & x_{pi}(k) & 0 & 1 & 0 \\ y_{pi}(k) & -x_{pi}(k) & 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{U}=\begin{bmatrix} x_{wi}-x_{pi}(k) \\ y_{wi}-y_{pi}(k) \\ z_{wi}-z_{pi}(k) \end{bmatrix} \quad (3)$$

$$\mathbf{U}=[\Delta R_x(k) \quad \Delta R_y(k) \quad \Delta R_z(k) \quad T_x \quad T_y \quad T_z]^T$$

In Eq.(3), i=1, 2, …, N, so there are a total number of 3N equations which form a linear equation group with respect to the vector $[\Delta R_x(k), \Delta R_y(k), \Delta R_z(k), T_x, T_y, T_z]^{\mathrm{T}}$ that can be obtained by solving the linear equation with least-square rule. Then $R_x(k+1)$, $R_z(k+1)$, $R_z(k+1)$ can be computed. After several rounds of iteration, R_x, R_y, R_z and T_x, T_y, T_z will converge; then the rotation matrix $\mathbf{R_{pw}}$ and the translation vector $\mathbf{T_{pw}}$ are obtained.

(b) The geometric transform relationship between the vehicle coordinate and the image coordinate

Suppose there are N control points and their coordinates in the vehicle coordinate and the image coordinate are respectively (x_{wi}, y_{wi}, z_{wi}) and (u_i, v_i); i=1, 2, … , N. The geometric transform between the vehicle coordinate and the image coordinate can be described by a perspective transform [6]:

$$\mathbf{M_{wf}}[x_{wi} \quad y_{wi} \quad z_{wi} \quad 1]^T=\beta[u_i \quad v_i \quad 1]^T \quad (4)$$

Where $\mathbf{M_{wf}}$ is the perspective matrix which can be computed by solving a linear equation group using the coordinate information of the control points [6]. It is worth noting that all the image coordinates mentioned in the paper are distortion removed. So the perspective transform shown in Eq.(4) works well.

3 Co-detection of pedestrians

The architecture of the co-detection method consists of four parts: 1) cluster and sift; 2) obtaining ROI; 3) obtaining contour edges; 4) decision. First, the range data are clustered and those clusters which might represent pedestrians (the object that such cluster represents is called candidate object) are sifted out; then proper ROI is obtained using the range data of candidate objects and the co-calibration results; after that edge-extraction is carried out in ROI and contour edges of candidate objects are obtained with the help of the range data of candidate objects and the co-calibration results; finally, decision on whether a candidate object is a pedestrian is made according to the feature of its contour edges.

3.1 Cluster and sift

The cluster rule is generally described as [7]: if $\|(x_{p,i}, y_{p,i})-(x_{p,i+1}, y_{p,i+1})\|<D_{thd}$, then the i-th and (i+1)-th scanning point of laser scanner are in the same cluster; otherwise they are in different clusters. Here, $D_{thd} = K_1\min\{\|(x_{p,i}, y_{p,i})\|, \|(x_{p,i+1}, y_{p,i+1})\|\}$, $K_1 = k\sin(\Delta\alpha/2)$; $\Delta\alpha$ is the angle resolution of laser scanner, k is a constant chosen according to experience.

Suppose there are totally N clusters C_1, C_2, …, C_N; for any cluster C_i: $\{(x_{p,n(i-1)}, y_{p,n(i-1)}), (x_{p,n(i-1)+1}, y_{p,n(i-1)+1}), \ldots, (x_{p,n(i)-1}, y_{p,n(i)-1})\}$, its diameter $d(C_i)$ is defined as: $d(C_i) = \max\{\|(x_{p,j}, y_{p,j})-(x_{p,l}, y_{p,l})\|: j,l\in[n(i-1), \ldots, n(i)-1]\}$; i=1, 2, …, N. Since the size of a pedestrian's body is limited, a range $[D_{min}, D_{max}]$ is set to sift the clusters according to their diameter. If and only if the diameter of a cluster is in this range, then this cluster is reserved, and the object that such cluster represents is called candidate object; otherwise this cluster is discarded. Above, set D_{min}=0.15m, D_{max}=1.2m.

3.2 Obtaining ROI

Given a candidate cluster C_i: $\{(x_{p,n(i-1)}, y_{p,n(i-1)}), (x_{p,n(i-1)+1}, y_{p,n(i-1)+1}), \ldots, (x_{p,n(i)-1}, y_{p,n(i)-1})\}$ ($z_p\equiv0$), its two end-points are $C_L(x_{p,n(i-1)}, y_{p,n(i-1)}, z_{p,n(i-1)})$ and $C_R(x_{p,n(i)-1}, y_{p,n(i)-1}, z_{p,n(i)-1})$. The coordinates of C_L and C_R can be computed using Eq.(1), denote them as $C_L^{\ w}(x_{w,n(i-1)}, y_{w,n(i-1)}, z_{w,n(i-1)})$ and $C_R^{\ w}(x_{w,n(i)-1}, y_{w,n(i)-1}, z_{w,n(i)-1})$. A pedestrian's body could be roughly represented by a vertical rectangular envelop, as shown in Fig2(a); the four corners are denoted as BC_1, BC_2, TC_1 and TC_2. Suppose $C_L^{\ w}$ and $C_R^{\ w}$ are respectively on the left and right side of the rectangular envelop; then the coordinates of these four corners in the vehicle coordinate are $BC_1(x_{w,n(i-1)}, y_{w,n(i-1)}, 0)$, $BC_2(x_{w,n(i)-1}, y_{w,n(i)-1}, 0)$, $TC_1(x_{w,n(i-1)}, y_{w,n(i-1)}, H_{max})$ and $TC_2(x_{w,n(i)-1}, y_{w,n(i)-1}, H_{max})$, where H_{max} denotes the height limit of human beings (set H_{max} = 2.5m). The rectangular envelop is expanded a bit (for example 30 cm) on both left side and right side.

The coordinates of the corresponding points of $C_L^{\ w}$, $C_R^{\ w}$, BC_1, BC_2, TC_1, TC_2 in the image coordinate can be computed using Eq.(4); their image coordinates are denoted respectively as $C_L^{\ I}(u_{CL}, v_{CL})$, $C_R^{\ I}(u_{CR}, v_{CR})$, $BC_1^{\ I}(u_{BC1}, v_{BC1})$, $BC_2^{\ I}(u_{BC2}, v_{BC2})$, $TC_1^{\ I}(u_{TC1}, v_{TC1})$, $TC_2^{\ I}(u_{TC2}, v_{TC2})$. The ROI should at least contain $BC_1^{\ I}$, $BC_2^{\ I}$, $TC_1^{\ I}$, $TC_2^{\ I}$; if a smallest vertical rectangle is chosen as the ROI, then its four corners are (u_{min}, v_{min}), (u_{min}, v_{max}), (u_{max}, v_{min}), (u_{max}, v_{max}); where $u_{min} = \min\{u_{BC1}, u_{BC2}, u_{TC1}, u_{TC2}\}$, $u_{max} = \max\{u_{BC1}, u_{BC2}, u_{TC1}, u_{TC2}\}$, $v_{min} = \min\{v_{BC1}, v_{BC2}, v_{TC1}, v_{TC2}\}$, $v_{max} = \max\{v_{BC1}, v_{BC2}, v_{TC1}, v_{TC2}\}$. As shown in Fig2, (b) shows the original image while (c) shows the extracted ROI of several candidate objects using above method.

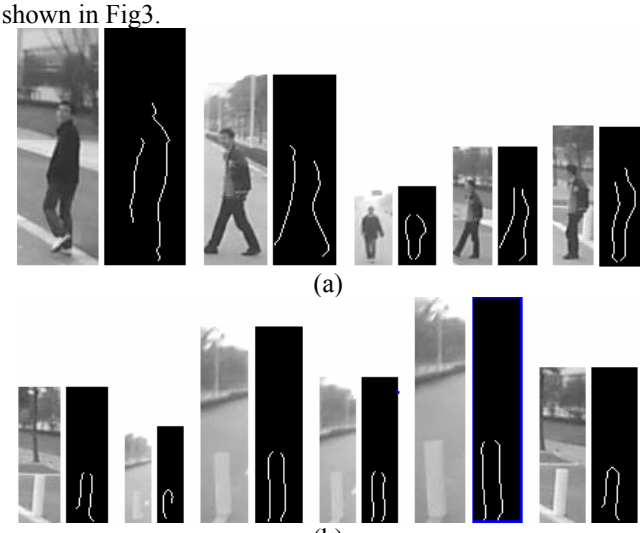
(a) (b) (c)

Fig 2 the rectangular envelope and the region of interest (ROI)

3.3 Obtaining contour edges

Edge-extraction is carried out in ROI using Canny method [8]. In ideal condition, $C_L^{\ I}(u_{CL}, v_{CL})$ and $C_R^{\ I}(u_{CR}, v_{CR})$ should be exactly on the left and right contour edge of candidate object. Actually, there will always be a slight deviation because of all kinds of errors. $\{(u_{CL}, v_{CL}), (u_{CR}, v_{CR})\}$

are matched with edge points in ROI using closest point or iterative closest point method, and two edge points on the left and right contour edges of candidate object are obtained. Then carry out edge point connection from these two edge points until discontinuity appears or the edge direction begins to deviate largely from vertical direction (on the consideration that the contour edges of a pedestrian are generally vertical). The extracted contour edges of several candidate objects are shown in Fig3.



(a) (b)

Fig 3 the contour edges of candidate objects

3.4 Decision

In this section, a decision rule is induced from a set of examples of contour shapes of candidate objects. Besides pedestrians themselves, the most likely candidate objects sifted from range data are landmarks, because of their similarity in size to pedestrians; as shown in Fig3. Therefore, the decision rule in some sense is a rule of how to distinguish pedestrians from landmarks.

A noticeable feature of landmarks is that their contour edges are almost straight on image, while the contour edges of pedestrians are irregular curved lines. A contour curve measure F_{curve} is defined as follows to describe the curved extent of contour edges of a candidate object.

Suppose the set of edge points on the left and on the right contour edge of the candidate object are respectively the set CL: $\{(u_{1,L}, v_{1,L}), (u_{2,L}, v_{2,L}), \ldots, (u_{m,L}, v_{m,L})\}$ and the set CR: $\{(u_{1,R}, v_{1,R}), (u_{2,R}, v_{2,R}), \ldots, (u_{n,R}, v_{n,R})\}$. Line l_{CL}: $a_{CL}u+b_{CL}v+c_{CL}=0$ and line l_{CR}:$a_{CR}u+b_{CR}v+c_{CR}=0$ are the straight lines fitted respectively to the edge points in set CL and in set CR with least square rule. The contour curve measure F_{curve} is defined as:

$$F_{curve}=\sqrt{\frac{\sum_{i=1}^{m} d((u_{i,L},v_{i,L}),l_{CL})^2+\sum_{j=1}^{n} d((u_{j,R},v_{j,R}),l_{CR})^2}{m+n}} \quad (5)$$

Where $d((u_{i,L}, v_{i,L}), l_{CL})$ denotes the distance between point $(u_{i,L}, v_{i,L})$ and line l_{CL}; $d((u_{j,R}, v_{j,R}), l_{CR})$ denotes the distance between point $(u_{j,R}, v_{j,R})$ and line l_{CR}.

A number of 500 samples of pedestrians and landmarks are chosen; the contour edges of each object are extracted and the contour curve measure is computed using Eq.(5). The statistic result of F_{curve} is shown in Fig4. It shows the probability distribution of the contour curve measure of sample pedestrians and landmarks. As it can be seen from Fig4 that there is an apparent dividing line (F_{curve}=1) between the distribution of pedestrians and landmarks. So the decision is induced as: if F_{curve}>1, then the candidate object is regarded as a pedestrian; otherwise, it is regarded as a landmark.

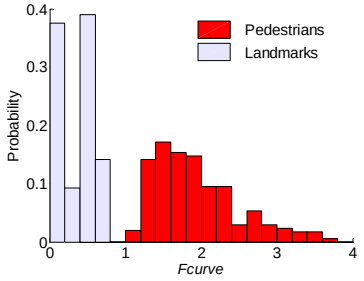


Fig 4 the probability distribution of the contour curve measure

4 Experiment

4.1 Experiment on the co-calibration method

Choose an arbitrary flat ground as the calibration field, as Fig5; put the calibration tool at several places in the calibration field and obtain the coordinate information of control points using the method introduced in section 2.2; compute the rotation matrix $\mathbf{R_{pw}}$ and the translation vector $\mathbf{T_{pw}}$ in Eq.(1) and the perspective matrix $\mathbf{M_{wf}}$ in Eq.(4) using the method introduced in section 2.3. Although the frame showed here happens to be roughly perpendicular to vehicle axis, it is not necessary to put the frame this way.

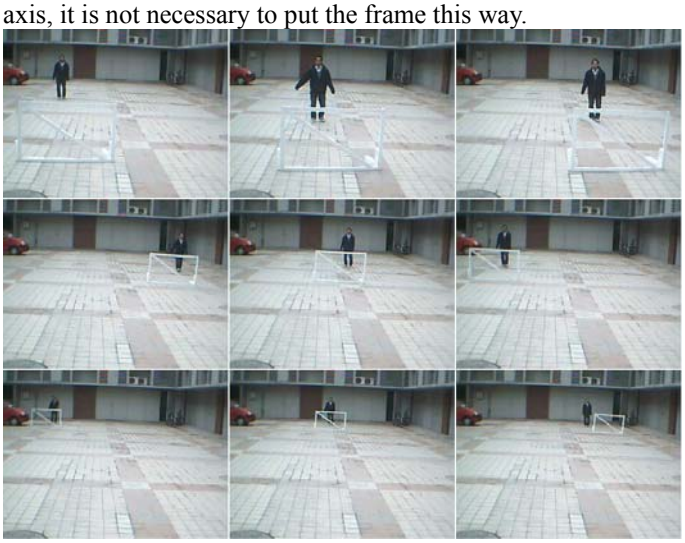


Fig 5 the calibration field

In order to examine the effect of the co-calibration method, each scanning point and part of laser beams are projected onto the image using the range data and the co-calibration results, thus forming a kind of augmented reality effect, as shown in Fig6. As it can be seen from Fig6, the projection matches well with the scenario, especially note that at each boundary of two neighboring objects, there is a

corresponding discontinuity in the projection of laser beam. Fig6 displays a kind of lifelikeness as if one can really see the laser beams and how they scan the scenario. Such lifelikeness indirectly testifies the effectiveness and accuracy of the proposed co-calibration method.

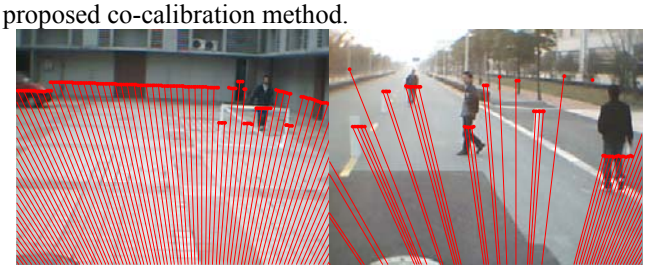


Fig 6 the projection of laser beam on the images

4.2 Experiment on the co-detection method

The intelligent vehicle used is the CyberC3 vehicle developed by the IV Lab of SJTU, with a 2D laser scanner SICK installed on the front of the vehicle and an off-the-shelf Logitech camera installed on the top of the vehicle. The camera and laser scanner are co-calibrated with the proposed method introduced in section2. The experiment scenario for pedestrian detection is shown in Fig7; the intelligent vehicle is moving on a road and its detection area is within 20m ahead and within 4m on two sides. Several pedestrians and landmarks appear in the detection area early or late during the experiment process. Every time the vision data and range data are recorded at the same time and the sample interval is about 0.1 second; every frame of recorded data is processed using the proposed co-detection method. Several images are displayed as example, shown in Fig7. A detected pedestrian is marked by bold yellow box while a detected landmark is marked by thin blue box, as shown in Fig7; it can be seen that the proposed method works well; the pedestrians as well as landmarks are correctly detected. During the whole experiment process, the omission detection ratio (omission detection means all the cases when the pedestrian is not detected or is detected as other object) and the false detection ratio (false detection means all the cases when a non-pedestrian is detected as a pedestrian) for pedestrians are respectively 1% and 3%.

Fig 7 the result of co-detection of pedestrians

5 Conclusion

In this paper, a new co-detection method using camera and laser scanner is proposed for pedestrian detection. First, a method of camera and laser scanner co-calibration is proposed, including how to obtain coordinate information of control points and how to compute the geometric transform relationship between the laser scanner coordinate, the vehicle coordinate and the image coordinate. Then a laser scanner and camera based pedestrian detection method using the co-calibration result is proposed, including how to sift candidate objects through range data, how to obtain proper ROI and contour edges of candidate objects using range data and co-calibration result, and how to make a decision on candidate object according to its contour feature. Experiments validate the efficiency of the proposed co-calibration method and co-detection method. In future work, more sophisticated image processing method such as SVM, NN, SIFT will integrated into the architecture of the proposed co-detection method.

Acknowledgement

This research is supported by the European CyberCars-2 project (FP6-028062) and the Shanghai Science Committee Foundation for the Mountaineering Action Plan (062107035).

Reference

[1] Broggi A, Bertozzi M, Fascioli A, Sechi M. Shape-based pedestrian detection. In: Proceeding of IEEE Intelligent Vehicles Symposium, Dearborn, USA. IEEE, 2000, pp.215~220

[2] Gavrila. D M, Pedestrian detection from a moving vehicle. In: Proceedings of the 6th European Conference on Computer Vision-Part II. London, UK: Springer-Verlag, 2000, pp.37~49

[3] Curio C, Edelbrunner J, Kalinker T, Tzomakas C, Wernervon Seelen, Walking pedestrian recognition, IEEE Transactions on Intelligent Transportation Systems, 2000,1(3): 155~163

[4] Shashua A, Gdalyahu Y, Hayun G, Pedestrian detection for driving assistance systems: single-frame classification and system level performance. In: Proceedings of IEEE Intelligent Vehicles Symposium, Parma, Italy, IEEE, 2004, pp.1~6

[5] Tons M, Doerfler R, Meinecke M M, Obojski M A, Radar sensors and sensors platform used for pedestrian protection in the EC-funded project SAVE-U. In: Proceedings of IEEE Intelligent Vehicles Symposium. Parma, Italy. IEEE, 2004, pp.813~818

[6] Zhang Z., Flexible camera calibration by viewing a plane from unknown orientations, in: Proceedings of 7th IEEE International Conference on Computer Vision, Corfu Greece, 1999, pp.666-673

[7] Cristiano Premebida, Urbano Nunes. Segmentation and geometric primitives extraction from 2D laser range data for mobile robot applications. Robotica 2005 – Actas do Encontro Cientifico, pp.17-25

[8] Canny J., A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(6): 679-698

Session II

Multi-sensor perception & navigation

- **Title: Model Based Vehicle Tracking in Urban Environments** *(invited paper)*
  Authors: Anna Petrovskaya and Sebastian Thrun

Model Based Vehicle Tracking in Urban Environments

Anna Petrovskaya and Sebastian Thrun
Computer Science Department
Stanford University
Stanford, California 94305, USA
{ anya, thrun }@cs.stanford.edu

***Abstract*— Situational awareness is crucial for autonomous driving in urban environments. We present the moving vehicle tracking module we developed for our autonomous driving robot Junior. The robot won second place in the Urban Grand Challenge, an autonomous driving race organized by the U.S. Government in 2007. The module provides reliable detection and tracking of moving vehicles from a high-speed moving platform using laser range finders. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them using a single Bayes filter per vehicle. We show how to build consistent and efficient 2D representations out of 3D range data and how to detect poorly visible black vehicles. Experimental validation includes the most challenging conditions presented at the Urban Grand Challenge as well as other urban settings.**

I. Introduction

Autonomously driving cars have been a long-lasting dream of robotics researchers and enthusiasts. Self-driving cars promise to bring a number of benefits to society, including prevention of road accidents, optimal fuel usage, comfort and convenience. In recent years the Defense Advanced Research Projects Agency (DARPA) has taken a lead on encouraging research in this area and organized a series of competitions for autonomous vehicles. In 2005 autonomous vehicles were able to complete a 131 mile course in the desert [1]. In the 2007 competition, the Urban Grand Challenge, the robots were presented with an even more difficult task: autonomous safe navigation in urban environments. In this competition the robots had to drive safely with respect to other robots, human-driven vehicles and the environment. They also had to obey the rules of the road as described in the California rulebook (see [2] for a detailed description of the rules). One of the most significant changes from the previous competition is the need for situational awareness of both static and dynamic parts of the environment. Our robot, Junior, won the second prize in the 2007 competition. An overview of Junior's software and hardware architecture is given in [3]. In this paper we describe the approach we developed for detection and tracking of moving vehicles.

Vehicle tracking has been studied for several decades. A number of approaches focused on the use of vision exclusively [4], [5], [6]. Whereas others utilized laser range finders [7], [8], [9] sometimes in combination with vision [10]. We give an overview of prior art in Sect. II.

For our application we are concerned with laser based vehicle tracking from the autonomous robotic platform Junior, to which we will also refer as the ego-vehicle (see Fig. 1). In contrast to prior art, we propose a model based approach which encompasses both geometric and dynamic

This work was in part supported by the Defense Advanced Research Projects Agency under contract number HR0011-06-C-0145. The opinions expressed in the paper are ours and not endorsed by the U.S. Government.

(a)

(b)

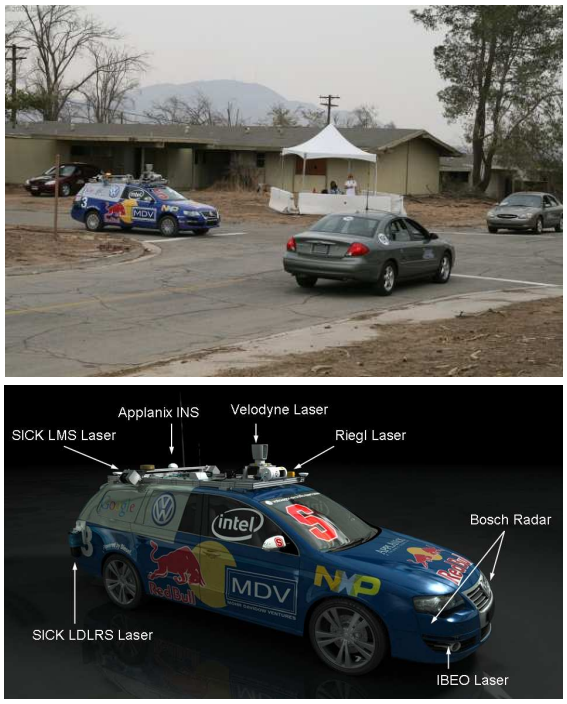
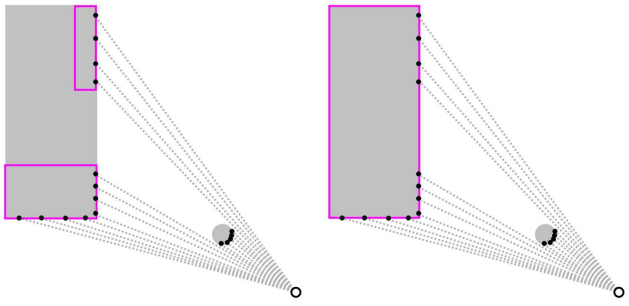
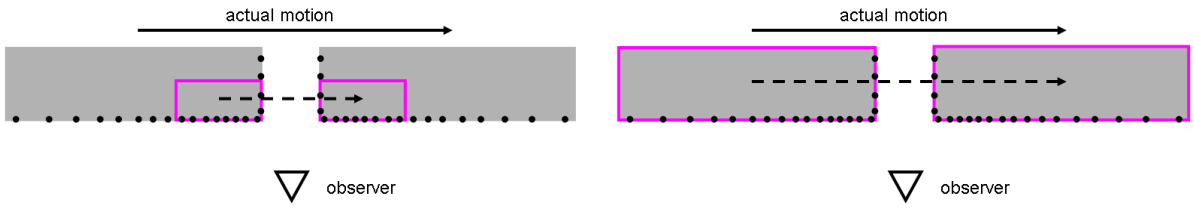


Fig. 1. (a) Our robot Junior (blue) negotiates an intersection with human-driven vehicles at the qualification event for the Urban Grand Challenge in November 2007. (b) Junior, is equipped with five different laser measurement systems, a multi-radar assembly, and a multi-signal inertial navigation system.



(a) without geometric model (b) with geometric model

Fig. 2. Scans from vehicles are often split up into separate clusters by occlusion. Geometric vehicle model helps interpret the data properly. Purple rectangles group together points that have been associated together. In (b) the purple rectangle also denotes the geometric vehicle model. Gray areas are objects. Gray dotted lines represent laser rays. Black dots denote laser data points. (Best viewed in color.)



(a) without shape estimation (b) with shape estimation

Fig. 3. Vehicles come in different sizes. Accurate estimation of geometric shape helps obtain a more precise estimate of the vehicle dynamics. Solid arrows show the actual distance the vehicle moved. Dashed arrows show the estimated motion. Purple rectangles denote the geometric vehicle models. Black dots denote laser data points. (Best viewed in color.)

properties of the tracked vehicle in a single Bayes filter. The approach eliminates the need for separate data segmentation and association steps. We show how to properly model the dependence between geometric and dynamic vehicle properties using *anchor point coordinates*. The geometric model allows us to naturally handle the disjoint point clusters that often result from partial occlusion of vehicles (see Fig. 2). Moreover, the estimation of geometric shape leads to accurate prediction of dynamic parameters (see Fig. 3).

Further, we introduce an abstract sensor representation, we call the *virtual scan*, which allows for efficient computation and can be used for a wide variety of laser sensors. We present techniques for building consistent virtual scans from 3D range data and show how to detect poorly visible black vehicles in laser scans. Our approach runs in real time with an average update rate of 40Hz, which is 4 times faster than the common sensor frame rate of 10Hz. The results show that our approach is reliable and efficient even in challenging traffic situations presented at the Urban Grand Challenge.

II. Background

Typically vehicle tracking approaches (e.g. [7], [8], [9], [10]) proceed in three stages: data segmentation, data association, and Bayesian filter update. During data segmentation the sensor data is divided into meaningful pieces (usually lines or clusters). During data association these pieces are assigned to tracked vehicles. Next a Bayesian filter update is performed to fit targets to the data.

The second stage - data association - is generally considered the most challenging stage of the vehicle detection and tracking problem because of the association ambiguities that arise. Typically this stage is carried out using variants of multiple hypothesis tracking (MHT) algorithm (e.g. [8], [9]). The filter update is usually carried out using variants of Kalman filter (KF), which is augmented by interacting multiple model method in some cases [7], [9].

Although vehicle tracking literature primarily relies on variants of KF, there is a great body of multiple target tracking literature for other applications (see [11] for a summary) where parametric, sample-based, and hybrid filters are used. For example [12] uses a Rao-Blackwellized particle filter (RBPF) for multiple target tracking on simulated data. A popular alternative to MHT for data association is the joint probabilistic data association (JPDA) method. For example in [13] a JPDA particle filter is used to track multiple targets from an indoor mobile robot platform.

The work included in this paper has been presented at two conferences: [14] and [15]. In contrast to prior vehicle tracking literature, we utilize a model based approach, which uses RBPFs and eliminates the need for separate data segmentation and association stages. Our approach estimates position, velocity and shape of tracked vehicles.

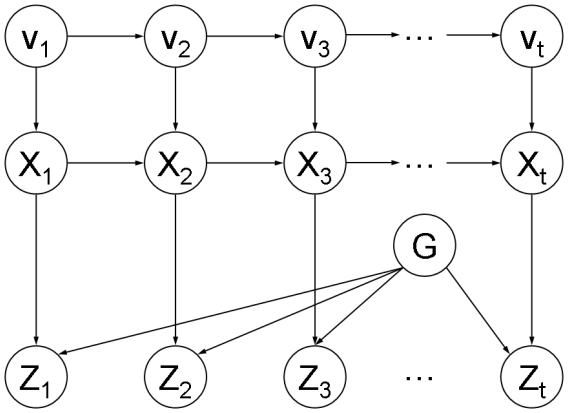


Fig. 4. Dynamic Bayesian network model of the tracked vehicle pose X_t, forward velocity v_t, geometry G, and measurements Z_t.

III. Representation

Our ego-vehicle is outfitted with the Applanix navigation system that provides pose localization with 1m accuracy. We further improved the localization module performance by observing lane markings [3]. Although global localization shifts may still occur, vehicle tracking is much more affected by localization drift rather than global shifts. For this reason we implemented *smooth coordinates*, which provide a locally consistent estimate of the ego-vehicle motion based on the data from the inertial measurement unit (IMU). As a result there is virtually no drift in the smooth coordinate system. Thus for the remainder of the paper we will assume that a reasonably precise pose of the ego-vehicle is always available.

Following the common practice in vehicle tracking, we will represent each vehicle with a separate Bayesian filter, and represent dependencies between vehicles via a set of local spatial constraints. Specifically we will assume that no two vehicles overlap, that all vehicles are spatially separated by some free space, and that all vehicles of interest are located on or near the road.

A. Probabilistic Model and Notation

For each vehicle we estimate its 2D position and orientation $X_t = (x_t, y_t, \theta_t)$ at time t, its forward velocity v_t and its geometry G (further defined in Sect. III-B). Also at each time step we obtain a new measurement Z_t. See

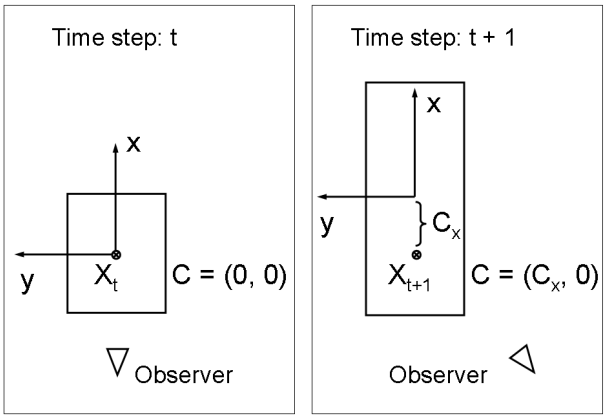


Fig. 5. As we move to observe a different side of a stationary car, our belief of its shape changes and so does the position of the car's center point. To compensate for the effect, we introduce local anchor point coordinates $C = (C_x, C_y)$ so that we can keep the anchor point X_t stationary in the world coordinates.

Fig. 4 for a dynamic Bayes network representation of the resulting probabilistic model. The dependencies between the parameters involved are modeled via probabilistic laws discussed in detail in Sects. III-C and III-E. For now we briefly note that the velocity evolves over time according to

$$p(v_t|v_{t-1}).$$

The vehicle moves based on the evolved velocity according to a dynamics model:

$$p(X_t|X_{t-1}, v_t).$$

The measurements are governed by a measurement model:

$$p(Z_t|X_t, G).$$

For convenience we will write $X^t = (X_1, X_2, ..., X_t)$ for the vehicle's trajectory up to time t. Similarly, v^t and Z^t will denote all velocities and all measurements up to time t.

B. Vehicle Geometry

The exact geometric shape of a vehicle can be complex and difficult to model precisely. For simplicity we approximate it by a rectangular shape of width W and length L. The 2D representation is sufficient because the height of the vehicles is not important for driving applications.

For vehicle tracking it is common to track the position of a vehicle's center within the state variable X_t. However, there is an interesting dependence between our belief about the vehicle's shape and position (Fig. 5). As we observe the object from a different vantage point, we change not only our belief of its shape, but also our belief of the position of its center point. Allowing X_t to denote the center point can lead to the undesired effect of obtaining a non-zero velocity for a stationary vehicle, simply because we refine our knowledge of its shape.

To overcome this problem, we view X_t as the pose of an *anchor point* who's position with respect to the vehicle's center can change over time. Initially we set the anchor point to be the center of what we believe to be the car shape and thus its coordinates in the vehicle's *local* coordinate system are $C = (0, 0)$. We assume that the vehicle's local coordinate system is tied to its center with the x-axis pointing directly forward. As we revise our knowledge of the vehicle's shape, the local coordinates of the anchor point will also need to be revised accordingly to $C = (C_x, C_y)$. Thus the complete set of geometric parameters is $G = (W, L, C_x, C_y)$.

C. Vehicle Dynamics Model

Given a vehicle's velocity v_{t-1} at time step $t-1$, the velocity evolves via addition of random bounded noise based on maximum allowed acceleration a_{max} and the time delay Δt between time steps $t-1$ and t. Specifically, we sample Δv uniformly from $[-a_{max}\Delta t,\ a_{max}\Delta t]$.

The pose evolves via linear motion - a motion law that is often utilized when exact dynamics of the object are unknown. The motion consists of perturbing orientation by $\Delta\theta_1$, then moving forward according to the current velocity by $v_t\Delta t$, and making a final adjustment to orientation by $\Delta\theta_2$. Again we sample $\Delta\theta_1$ and $\Delta\theta_2$ uniformly from $[-d\theta_{max}\Delta t,\ d\theta_{max}\Delta t]$ for a maximum allowed orientation change $d\theta_{max}$.

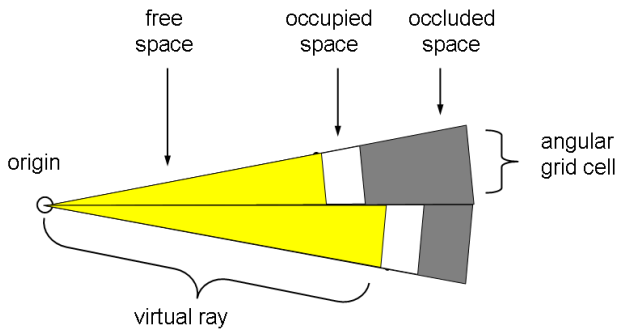
D. Sensor Data Representation

In this paper we focus on laser range finders for sensing the environment. Recently these sensors have evolved to be more suitable for driving applications. For example IBEO Alasca sensors allow for easy ground filtering by collecting four parallel horizontal scan lines and marking which of the readings are likely to come from the ground. Velodyne HDL-64E sensors do not provide ground filtering, however they take a 3D scan of the environment at high frame rates (10Hz) thereby producing 1,000,000 readings per second. Given such rich data, the challenge has become to process the readings in real time. Vehicle tracking at 10 - 20Hz is desirable for driving decision making.

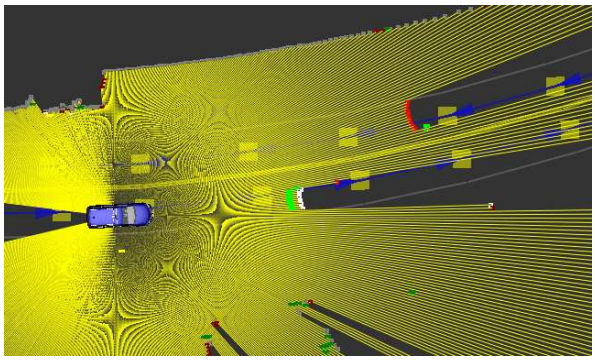
A number of factors make the use of raw sensor data inefficient. As the sensor rotates to collect the data, each new reading is made from a new vantage point due to ego-motion. Ignoring this effect leads to significant sensor noise. Taking this effect into account makes it difficult to quickly access data that pertains to a specific region of space. Much of the data comes from surfaces uninteresting for the purpose of vehicle tracking, e.g. ground readings, curbs and tree tops. Finally, the raw 3D data wastes a lot of resources as vehicle tracking is a 2D application where the cars are restricted to move on the ground surface. Therefore it is desirable to pre-process the data to produce a representation tailored for vehicle tracking.

To expedite computations, we construct a grid in polar coordinates - a *virtual scan* - which subdivides 360° around a chosen origin point into angular grids (see Fig. 6). In each angular grid we record the range to the closest obstacle. Hence each angular grid contains information about free, occupied, and occluded space. We will often refer to the cone of an angular grid from the origin until the recorded range as a *ray* due to its similarity to a laser ray.

Virtual scans simplify data access by providing a single point of origin for the entire data set, which allows constant time look-up for any given point in space. As we mentioned earlier it is important to compute correct world coordinates for the raw sensor readings. However, once the correct positions of obstacle points have been computed, adjusting the origin of each ray to be at the common origin for the virtual scan produces an acceptable approximation. Constructed in this manner, a virtual scan provides a compact representation of the space around the ego-vehicle classified

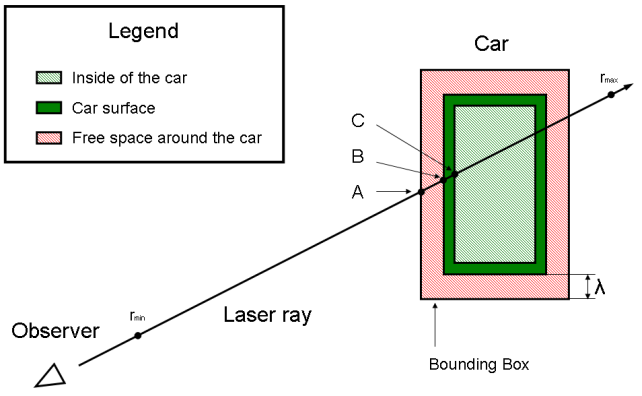


(a) anatomy of a virtual scan

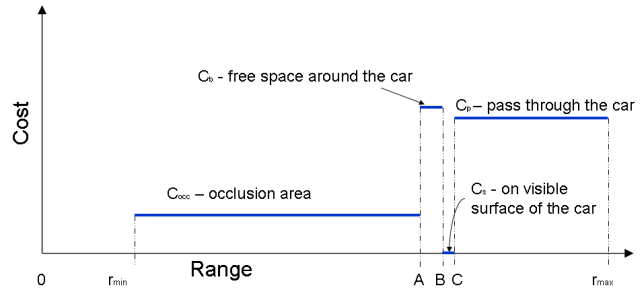


(b) a virtual scan constructed from Velodyne data

Fig. 6. In (b) yellow line segments represent virtual rays. Colored points show the results of a scan differencing operation. Red points are new obstacles, green points are obstacles that disappeared, and white points are obstacles that remained unchanged or appeared in previously occluded areas. (Best viewed in color.)



(a)



(b)

Fig. 7. Measurement likelihood computations. (a) shows the geometric regions involved in the likelihood computations. (b) shows the costs assignment for a single ray. (Best viewed in color.)

into free, occupied and occluded. The classification helps us properly reason about what parts of an object should be visible as we describe in Sect. III-E.

For the purpose of vehicle tracking it is crucial to determine what changes take place in the environment over time. With virtual scans these changes can be easily computed in spite of the fact that ego-motion can cause two consecutive virtual scans to have different origins. The changes are computed by checking which obstacles in the old scan are cleared by rays in the new scan and vice versa. This computation takes time linear in the size of the virtual scan and only needs to be carried out once per frame. Fig. 6(b) shows results of a virtual scan differencing operation with red points denoting new obstacles, green points denoting obstacles that disappeared, and white points denoting obstacles that remained in place or appeared in previously occluded areas.

Virtual scans are a suitable representation for a wide variety of laser range finders. While this representation is easy to build for 2D sensors such as IBEO, for 3D range sensors additional considerations are required to produce consistent 2D representations. We describe these techniques in Sect. V.

E. Measurement Model

Given a vehicle's pose X, geometry G and a virtual scan Z we compute the measurement likelihood $p(Z|G,X)$ as follows. We position a rectangular shape representing the vehicle according to X and G. Then we build a bounding box to include all points within a predefined distance λ[1] around the vehicle (see Fig. 7). For an actual vehicle in this configuration, we would expect the points within the rectangle to be occupied or occluded, and points in its vicinity to be free or occluded, because vehicles are spatially separated from other objects in the environment.

Following the common practice for modeling laser range finders, we consider measurements obtained along each ray independent of each other. Thus if we have a total of N rays in the virtual scan Z, the measurement likelihood factors as follows:

$$p(Z|G,X) = \prod_{i=1}^{N} p(z_i|G,X).$$

We model each ray's likelihood as a zero-mean Gaussian of variance σ_i computed with respect to a cost c_i selected based on the relationship between the ray and the vehicle (η_i is a normalization constant):

$$P(z_i|G,X) = \eta_i \ \exp\{ -\frac{c_i^2}{\sigma_i^2} \}.$$

The costs and variances are set to constants that depend on the region in which the reading falls into (see Fig. 7 for illustration). c_{occ}, σ_{occ} are the settings for range readings that fall short of the bounding box and thus represent situations when another object is occluding the vehicle. c_b and σ_b are the settings for range readings that fall short of the vehicle but inside of the bounding box. c_s and σ_s are the settings

[1]We used the setting of $\lambda = 1m$ in our implementation.

for readings on the vehicle's visible surface (that we assume to be of non-zero depth). c_p, σ_p are used for rays that extend beyond the vehicle's surface.

The domain for each range reading is between minimum range r_{min} and maximum range r_{max} of the sensor. Since the costs we select are piece-wise constant, it is easy to integrate the unnormalized likelihoods to obtain the normalization constants η_i. Note that for the rays that do not target the vehicle or the bounding box, the above logic automatically yields uniform distributions as these rays never hit the bounding box.

Note that the proposed measurement model naturally handles partially occluded objects including objects that are "split up" by occlusion into several point clusters (see Fig. 2). In contrast these cases are often challenging for approaches that utilize separate data segmentation and correspondence methods.

IV. Vehicle Tracking

Most vehicle tracking methods described in the literature apply separate methods for data segmentation and correspondence matching before fitting model parameters via extended Kalman filter (EKF). In contrast we use a single Bayesian filter to fit model parameters from the start. This is possible because our model includes both geometric and dynamic parameters of the vehicles and because we rely on efficient methods for parameter fitting. We chose the particle filter method for Bayesian estimation because it is more suitable for multi-modal distributions than EKF. Unlike the multiple hypothesis tracking (MHT) method commonly used in the literature, the computational complexity for our method grows linearly with the number of vehicles in the environment, because vehicle dynamics dictates that vehicles can only be matched to data points in their immediate vicinity. The downside of course is that in our case two targets can in principle merge into one. In practice we have found that it happens rarely and only in situations where one of the targets is lost due to complete occlusion. In these situations target merging is acceptable for our application.

We have a total of eight parameters to estimate for each vehicle: $X = (x, y, \theta)$, v, $G = (W, L, C_x, C_y)$. Computational complexity grows exponentially with the number of parameters for particle filters. Thus to keep computational complexity low, we turn to RBPFs first introduced in [16]. We estimate X and v by samples and keep Gaussian estimates for G within each particle. Below we give a brief derivation of the required update equations.

A. Update Equations

At each time step t we produce an estimate of a Bayesian belief about the tracked vehicle's trajectory, velocity and geometry based on a set of measurements:

$$Bel_t = p(X^t, v^t, G|Z^t).$$

We split up the belief into two conditional factors:

$$Bel_t = p(X^t, v^t|Z^t) \;\; p(G|X^t, v^t, Z^t).$$

The first factor encodes the vehicle's motion posterior:

$$R_t = p(X^t, v^t|Z^t).$$

The second factor encodes the vehicle's geometry posterior, conditioned on its motion:

$$S_t = p(G|X^t, v^t, Z^t).$$

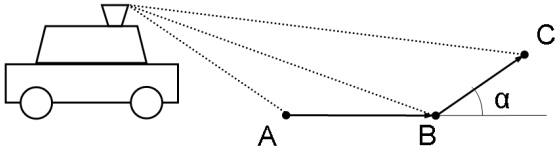


Fig. 8. We determine ground readings by comparing angles between consecutive readings. If A, B, C are ground readings, then α is close to 0 and thus $\cos\alpha$ is close to 1.

The factor R_t is approximated using a set of particles; the factor S_t is approximated using a Gaussian distribution (one Gaussian per particle).

Detailed derivations of the update equations are provided in [15]. Here we briefly note that the motion update of the particle filter is carried out using the vehicle dynamics model described in Sect. III-C. The measurement update is carried out by computing the importance weights w_t for all particles:

$$w_t = I\!E_{S_{t-1}}[\; p(Z_t|G, X_t) \;].$$

In words, the importance weights are the expected value (with respect to the vehicle geometry prior) of the measurement likelihood. Using Gaussian approximations of the geometry prior S_{t-1} and the measurement likelihood $p(Z_t|G, X_t)$, this expectation can be computed in closed form. We obtain a Gaussian approximation of the geometry prior recursively and apply Laplace's method to approximate the measurement likelihood by a Gaussian.

B. Initializing and Discontinuing Tracks

New tracks are initialized in areas where scan differencing detects a change in data, that is not already explained by existing tracks. New tracks are fitted using the same measurement and motion models (Sects. III-E and III-C) that we use for vehicle tracking. The candidates are vetted for three frames before they can become "real tracks". Detection of new vehicles is the most computationally expensive part of vehicle tracking. In order to achieve reliable vehicle detection in real time, we developed a number of optimization techniques. Details of the detection algorithm and optimizations can be found in [14].

We discontinue tracks if the target vehicle gets out of sensor range or moves too far away from the road[2]. We also discontinue tracks if the unnormalized weights have been low for several turns. Low unnormalized weights signal that the sensor data is insufficient to track the target, or that our estimate is too far away from the actual vehicle. This logic keeps the resource cost of tracking occluded objects low, yet it still allows for a tracked vehicle to survive bad data or complete occlusion for several turns. Since new track acquisition only takes three frames, it does not make sense to continue tracking objects that are occluded for significantly longer periods of time.

V. Working with 3D Range Data

As we explained in Sect. III-D, vehicle tracking is a 2D problem, for which compact 2D virtual scans are sufficient. However for 3D sensors, such as Velodyne, it is non-trivial to build consistent 2D virtual scans. These sensors provide immense 3D data sets of the surroundings, making

[2]A digital street map was available for our application in the Road Network Definition Format (RNDF).

computational efficiency a high priority when processing the data. In our experience, the hard work pays off and the resulting virtual scans carry more information than 2D sensor data.

A. Classification of 3D Points

To produce consistent 2D virtual scans, we need to understand which of the 3D data points should be considered obstacles. From the perspective of driving applications we are interested in the slice of space directly above the ground and about 2m high, as this is the space that a vehicle would actually have to drive through. Objects elevated more than 2m above ground - e.g. tree tops or overpasses - are not obstacles. The ground itself is not an obstacle (assuming the terrain is drivable). Moreover, for tracking applications low obstacles such as curbs should be excluded from virtual scans, because otherwise they can prevent us from seeing more important obstacles beyond them. The remaining objects in the 2m slice of space are obstacles for a vehicle, even if these objects are not directly touching the ground.

In order to classify the data into the different types of objects described above we first build a 3D grid in spherical coordinates. Similarly to a virtual scan, it has a single point of origin and stores actual world coordinates of the sensor readings. Just as in the 2D case, this grid is an approximation of the sensor data set, because the actual laser readings in a scan have varying points of origin. In order to downsample and reject outliers, for each spherical grid cell we compute the median range of the readings falling within it. This gives us a single obstacle point per grid cell. For each spherical grid cell we will refer to the cone from the grid origin to the obstacle point as a virtual ray.

The first classification step is to determine ground points. For this purpose we select a single slice of vertical angles from the spherical grid (i.e. rays that all have the same bearing angle). We cycle through the rays in the slice from the lowest vertical angle to the highest. For three consecutive readings A, B, and C, the slope between AB and BC should be near zero if all three points lie on the ground (see Fig. 8 for illustration). If we normalize AB and BC, their dot product should be close to 1. Hence a simple thresholding of the dot product allows us to classify ground readings and to obtain estimates of local ground elevation. Thus one useful piece of information we can obtain from 3D sensors is an estimate of ground elevation.

Using the elevation estimates we can classify the remaining non-ground readings into low, medium and high obstacles, out of which we are only interested in the medium ones (see Fig. 9). It turns out that there can be medium height obstacles that are still worth filtering out: birds, insects and occasional readings from cat-eye reflectors. These obstacles are easy to filter, because the BC vector tends to be very long (greater than 1m), which is not the case for normal vertical obstacles such as buildings and cars. After identifying the interesting obstacles we simply project them on the 2D horizontal plane to obtain a virtual scan.

B. Detection of Black Obstacles

Laser range finders are widely known to have difficulty seeing black objects. Since these objects absorb light, the sensor never gets a return. Clearly it is desirable to "see" black obstacles for driving applications. Other sensors could be used, but they all have their own drawbacks. Here we present a method for detecting black objects in 3D laser data. Figure 10 shows the returns obtained from a black car. The only readings obtained are from the license plate and wheels of the vehicle, all of which get filtered out as low obstacles. Instead of looking at the little data that is present, we can detect the black obstacle by looking at the data that is absent. If no readings are obtained along a range of vertical angles in a specific direction, we can conclude that the space must be occupied by a black obstacle. Otherwise the rays would have hit some obstacle or the ground. To provide a conservative estimate of the range to the black obstacle we place it at the last reading obtained in the vertical angles just before the absent readings. We note that this method works well as long as the sensor is good at seeing the ground. For the Velodyne sensor the range within which the ground returns are reliable is about 25 - 30m, beyond this range the black obstacle detection logic does not work.

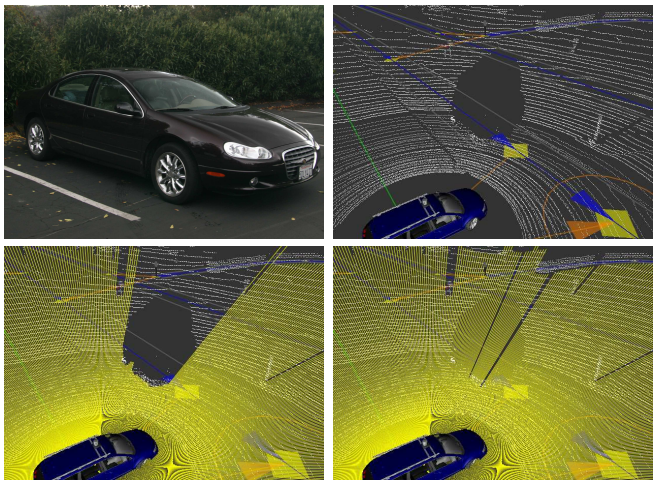
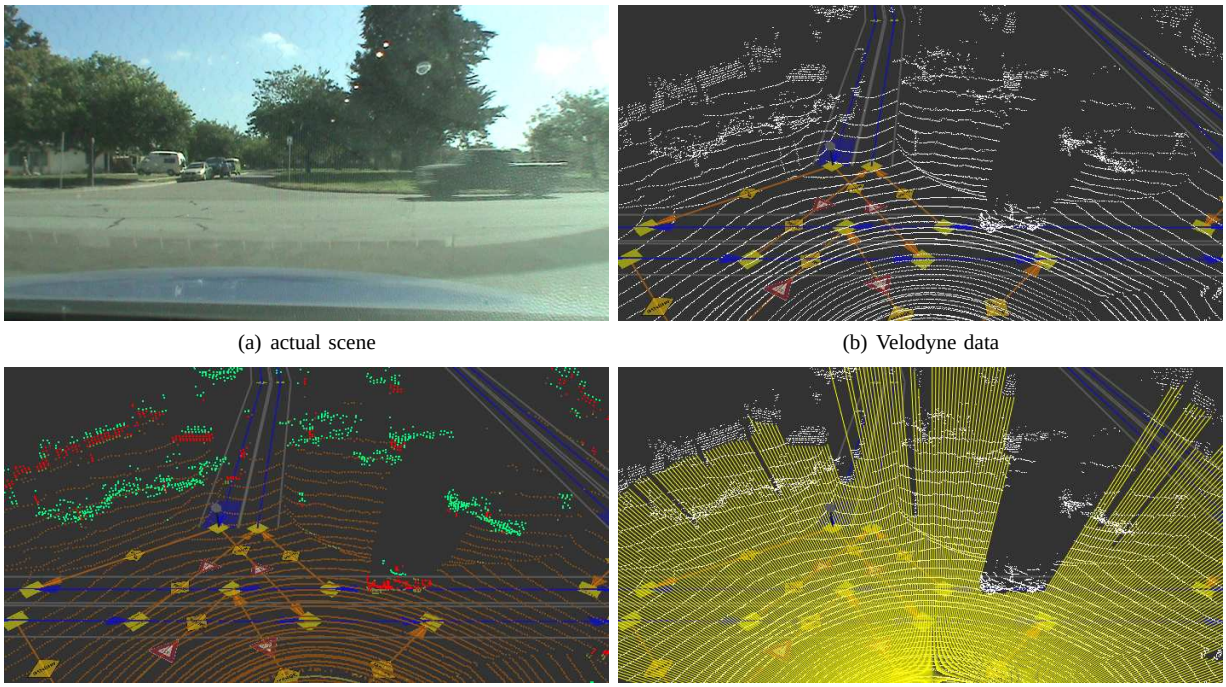


Fig. 10. Detecting black vehicles in 3D range scans. White points represent raw Velodyne data. Yellow lines represent the generated virtual scans. Top left: actual appearance of the vehicle. Top right: the vehicle gives very few laser returns. Bottom left: virtual scan with black object detection. Bottom right: virtual scan without black object detection.

VI. Experimental Validation

The most challenging traffic situation at the Urban Grand Challenge was presented on course A during the qualifying event (Fig. 11) . The test consisted of dense human driven traffic in both directions on a course with an outline resembling the Greek letter θ. The robots had to merge repeatedly into the dense traffic. The merge was performed using a left turn, so that the robots had to cross one lane of traffic each time. In these conditions accurate estimates of positions and velocities of the cars are very useful for determining a gap in traffic large enough to perform the merge safely. Cars passed in close proximity to each other and to stationary obstacles (e.g. signs and guard rails) providing plenty of opportunity for false associations. Partial and complete occlusions happened frequently due to the traffic density. Moreover these occlusions often happened near merge points which complicated decision making.

During extensive testing, the performance of our vehicle tracking module has been very reliable and efficient (see Fig. 11). Geometric shape of vehicles was properly estimated (see Figs. 12 and 13), which increased tracking reliability and improved motion estimation. The tracking approach proved



(a) actual scene (b) Velodyne data

(c) after classification (d) generated virtual scan

Fig. 9. In (c) Velodyne data is colored by type: orange - ground, yellow - low obstacle, red - medium obstacle, green - high obstacle. In (d) yellow lines denote the virtual scan. Note the truck crossing the intersection, the cars parked on a side of the road and the white van parked on a driveway. On the virtual scan all of these vehicles are clearly marked as obstacles, but ground, curbs and tree tops are ignored.

TABLE I

TRACKER PERFORMANCE ON DATA SETS FROM THREE URBAN ENVIRONMENTS. MAX TP IS THE THEORETICALLY MAXIMUM POSSIBLE TRUE POSITIVE PERCENT FOR EACH DATA SET. TP AND FP ARE THE ACTUAL TRUE POSITIVE AND FALSE POSITIVE RATES ATTAINED BY THE ALGORITHM.

Data Sets	Total Frames	Total Vehicles	Correctly Identified	Falsely Identified	Max TP (%)	TP (%)	FP (%)
UGC Area A	1,577	5,911	5,676	205	97.8	96.02	3.35
Stanford Campus	2,140	3,581	3,530	150	99.22	98.58	4.02
Alameda Day 1	1,531	901	879	0	98.22	97.56	0
Overall	5,248	10,393	10,085	355	98.33	97.04	3.3

capable of handling complex traffic situations such as the one presented on course A of the UGC. The computation time of our approach averages at 25ms per frame, which is faster than real time for most modern laser range finders.

We also gathered empirical results of the tracking module performance on data sets from several urban environments: course A of the UGC, Stanford campus and a port town in Alameda, CA. For each frame of data we counted how many vehicles a human is able to identify in the laser range data. The vehicles had to be within 50m of the ego-vehicle, on or near the road, and moving with a speed of at least 5mph. We summarize the tracker's performance in Tbl. I. Note that the maximum theoretically possible true positive rate is lower than 100% because three frames are required to detect a new vehicle. On all three data sets the tracker performed very close to the theoretical bound. Overall the true positive rate was 97% compared to the theoretical maximum of 98%.

Several videos of vehicle detection and tracking using the techniques presented in this paper are available at the website

http://cs.stanford.edu/people/petrovsk/uc.html

VII. CONCLUSIONS

We have presented the vehicle tracking module developed for Stanford's autonomous driving robot Junior. Tracking is performed from a high-speed moving platform and relies on laser range finders for sensing. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them with a single Bayes filter per vehicle. In contrast to prior art, the common data segmentation and association steps are carried out as part of the filter itself. The approach has proved reliable, efficient and capable of handling challenging traffic situations, such as the ones presented at the Urban Grand Challenge.

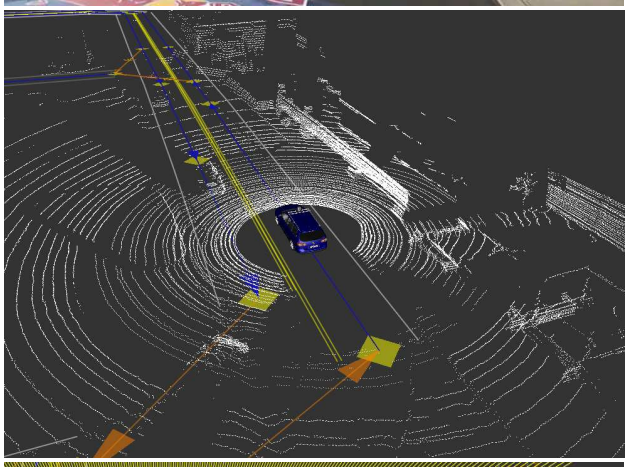
Clearly there is ample room for future work. The presented approach does not model pedestrians, bicyclists, or motorcyclists, which is a prerequisite for driving in populated areas. Another promising direction for future work is fusion of different sensors, including laser, radar and vision.

VIII. ACKNOWLEDGEMENTS

This research has been conducted for the Stanford Racing Team and would have been impossible without the whole team's efforts to build the hardware and software that makes up the team's robot Junior. The authors thank all team

(a)

(b)



(c)

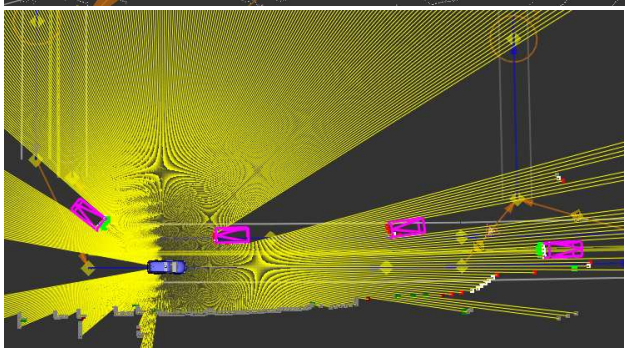


Fig. 11. Tracking results on course A at the UGC. (a) actual scene, (b) Velodyne data, (c) virtual scan and tracking results. In (c) yellow line segments represent the virtual scan and red/green/white points show results of scan differencing. The purple boxes denote the tracked vehicles. (Best viewed in color.)

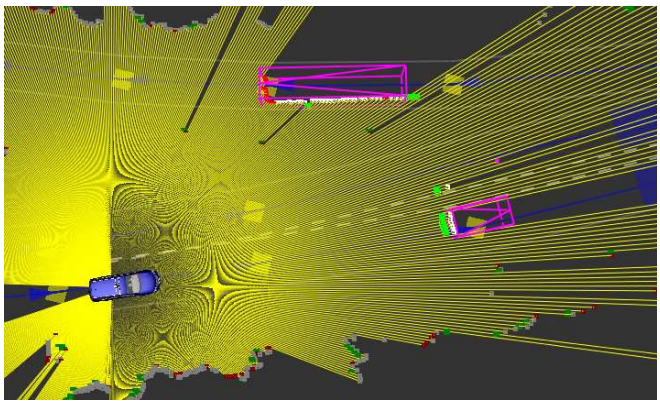
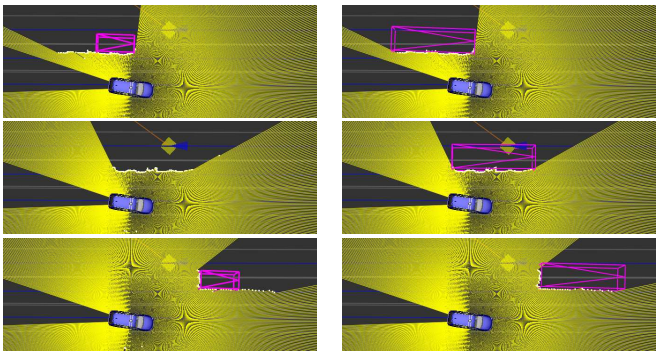


Fig. 12. Size estimation results on Stanford campus. Vehicles of different sizes are successfully estimated and tracked. (Best viewed in color.)



(a) without size estimation (b) with size estimation

Fig. 13. Size estimation improves accuracy of tracking as can be seen on the example of a passing bus taken from an Alameda data set. Without size estimation (a) the tracking results are poor because the geometric model does not fit the data well. Not only is the velocity estimated incorrectly, but the track is lost entirely when the bus is passing. With size estimation (b) the bus is tracked successfully and the velocity is properly estimated. (Best viewed in color.)

members for their hard work. The Stanford Racing Team is indebted to DARPA for creating the Urban Challenge, and for its financial support under the Track A Program. Further, Stanford University thanks its various sponsors. Special thanks also to NASA Ames for permission to use their air field.

REFERENCES

[1] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 Darpa Grand Challenge: The Great Robot Race*. Springer Verlag, 2007.
[2] DARPA, *Urban challenge rules, revision oct. 27, 2007*. See www.darpa.mil/grandchallenge/rules.asp, 2007.
[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.
[4] T. Zielke, M. M. Brauckmann, and W. v. Seelen, "Intensity and edge based symmetry detection applied to car following," in *ECCV, Berlin, Germany*, 1992.
[5] E. Dickmanns, "Vehicles capable of dynamic vision," in *IJCAI, Nagoya, Japan*, 1997.
[6] F. Dellaert and C. Thorpe, "Robust car tracking using kalman filtering and bayesian templates," in *Conference on Intelligent Transportation Systems*, 1997.
[7] L. Zhao and C. Thorpe, "Qualitative and quantitative car tracking from a range image sequence," in *Computer Vision and Pattern Recognition*, 1998.
[8] D. Streller, K. Furstenberg, and K. Dietmayer, "Vehicle and object models for robust tracking in traffic scenes using laser range images," in *Intelligent Transportation Systems*, 2002.
[9] C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research, Sep 2007; vol. 26: pp. 889-916*, 2007.
[10] S. Wender and K. Dietmayer, "3d vehicle detection using a laser scanner and a video camera," in *6th European Congress on ITS, Aalborg, Denmark*, 2007.
[11] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE AE Systems Magazine*, 2004.
[12] Särkkä, S. and Vehtari, A. and Lampinen, J., "Rao-blackwellized particle filter for multiple target tracking," *Inf. Fusion*, vol. 8, no. 1, 2007.
[13] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *ICRA*, 2001.
[14] A. Petrovskaya and S. Thrun, "Efficient techniques for dynamic vehicle detection," in *ISER, Athens, Greece*, 2008.
[15] A. Petrovskaya and S. Thrun, "Model based vehicle tracking for autonomous driving in urban environments," in *RSS, Zurich, Switzerland*, 2008.
[16] A. Doucet, N. d. Freitas, K. Murphy, and S. Russell, "Rao-blackwellised filtering for dynamic bayesian networks," in *UAI, San Francisco, CA*, 2000.

Session II

Multi-sensor perception & navigation

- **Title: Connexity based fronto-parallel plane detection for stereovision obstacle segmentation**
  Authors: Thomas Veit

Connexity based fronto-parallel plane detection for stereovision obstacle segmentation

Thomas Veit

***Abstract*— Progress in hardware makes it possible to compute dense disparity maps in real-time. This work describes a suitable obstacle segmentation method for these dense disparity maps. The method analyses the connexity of the disparity map in order to extract fronto-parallel planes by means of a suitable depth constraint. This pragmatic geometrical approach reduces the number of detection parameters. As a consequence it is easy and intuitive without requiring expert knowledge of the segmentation algorithm. The target application field is Advanced Driving Assistance Systems (ADAS). The performance of the method is illustrated by various results on real image sequences in the context of pedestrian detection.**

I. Introduction

The development of Advance Driving Assistance Systems (ADAS) necessitates tools that are able to interpret the surroundings of a vehicle. As an example, a path planning algorithm needs to compute the free navigable space in order to determine which direction the vehicle should take. The issue addressed in this paper is part of the process of analysing a road scene. The proposed method intends to extract from the scene the objects that are relevant for its interpretation.

The aim of this work is to segment the disparity map obtained from a stereo-vision system. In the field of intelligent vehicles, stereo-vision is backed up as an alternative to high cost laser sensors by the real-time computation of dense disparity maps [1]. Compared to 1D scanning sensors, stereo-vision not only provides depth information but also height on top of the full intensity information of the scene.

The outline of this paper is the following. Section II presents some related work. The computation of the disparity map is briefly described in Section III. The obstacle segmentation process is detailed in Section IV. Section V presents some experimental results in the context of pedestrian detection. Finally, Section VI gives some concluding remarks.

II. Related work

A large amount of literature discusses the issue of stereo-vision obstacle detection. The problem is studied from different angles. In [2] a method is proposed that extracts obstacles without computing a disparity map. The system analyses polar histograms obtained after compensating the perspective in the left and right images. This work focuses more on free-space estimation than on obstacle segmentation.

The authors of [3] present a framework for road plane estimation, marker-less road segmentation and obstacle segmentation. The obstacle segmentation step is based on a split and merge technique in order to group pixels with similar disparities.

The problem of obstacle detection is addressed in [4] by projecting the 3-dimensional disparity data on a 2D plane where obstacles appear as line segments. Thus, the obstacles extraction problem is translated into a line extraction problem making extensive use of the Hough transform. Unfortunately, the line segment model for obstacle is only an approximation and the line detection in the 2D space still turns out to be complex due to low contrast and discontinuities.

The work in [5] is the closest to the one proposed in this paper. The authors rely on a watershed segmentation in order to group disparity values into regions corresponding to obstacles. This watershed segmentation is applied on a multi-scale morphological gradient image of the disparity map.

III. Disparity map computations

The obstacle detection algorithm relies on a classical stereo-vision framework. The images acquired by the left and right camera are supposed to be rectified (i.e. epipolar lines correspond in both images). The disparity map is computed by applying a Sum of Absolute Difference (SAD) matching cost on squared aggregation windows and a winner-take-all strategy. The size of the windows is chosen rather large (9×9 for 320×240 images). Indeed, the foreground fattening effect is beneficial for obstacle detection. The disparity map is computed after filtering the input images with a Laplacian filter in order to be robust against global illumination changes between both cameras. In order to reduce the number of errors in the disparity map due to false correspondences, a symmetry constraint is enforced: pixels for which the disparity values for left to right and right to left matching differ are discarded. Finally, the disparity map is post-processed with a 3×3 median filter in order to enhance spatial coherence and reduce errors due to noise. An example of the resulting disparity map is presented in Fig. 1.

IV. Obstacle segmentation

The proposed obstacle segmentation algorithm proceeds in three steps. First, the road plane is suppressed.

T. Veit is with LIVIC / INRETS, 14 route de la Minière, F-78000 Versailles France, Thomas.Veit@inrets.fr

Fig. 1. Left image, right image and corresponding disparity map

Then, connected components in the disparity map are extracted according to a specific depth constraint. Finally, the connected components satisfying a set of geometrical constraint on their height, width and position are selected. Fig. 2 sums up the general flow of the algorithm.

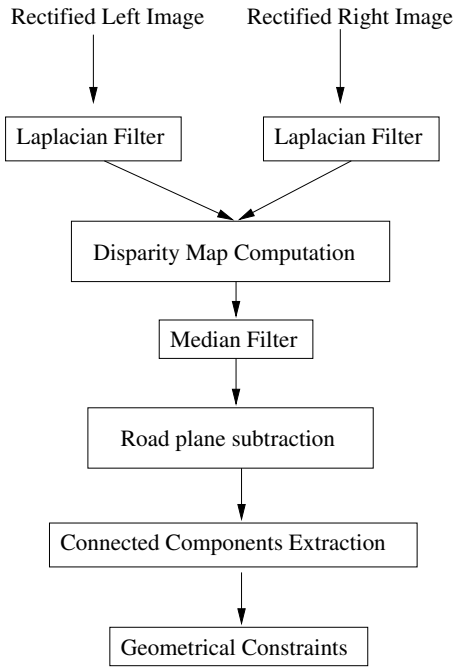


Fig. 2. General flow of the algorithm

A. Road plane subtraction

The first step of the detection algorithm is to suppress from the disparity map the values that correspond to the road plane. Indeed the proposed detection method relies on the classical assumption that the obstacle of interest are above the ground plane. If the position of the camera with respect to the road plane is assumed to be known then the disparity $d_{\max}(v)$ of a pixel (u,v) belonging to the road plane is easily related to its vertical image position:

$$d_{\max}(v) = ((v - v_0)\cos\theta + \alpha\sin\theta)\frac{b}{h}, \qquad (1)$$

where α is the focal length expressed in pixels, v_0 is the vertical image position of the optical center, b is the baseline distance of the stereo-system, h is the height of the camera with respect to the road plane and θ is the angle between the optical axes and the road plane.

All pixels which, for a given vertical image position, have a disparity that is lower than the road plane (in other words all points below the road plane) are discarded from further processing. In practice, a slightly stronger constraint is enforced: all points that are below a planar surface that is 0.3m above the road plane are discarded. This margin enables to deal with situations where a sidewalk is present or where the planar road assumption is only partly verified.

Now, the position of the camera with respect to the road plane can either be assumed constant or it can be dynamically estimated. The first alternative is suitable for smooth urban driving conditions and situations were acceleration and deceleration are reasonably low. However, when the vehicle's dynamics causes strong variations of the camera pitch angle, it has to be reestimated using methods such as the Hough transform [4], least-squares [3] or RANSAC [5]. More sophisticated techniques estimating a non-planar road surface might also be applied.

B. Connected component extraction

Once the values that correspond to the road surface are removed it is possible to focus on obstacles. One major characteristic of obstacles is that they can be approximated as fronto-parallel surfaces with respect to the image plane. In other words, obstacles are represented by approximately constant connected regions in a disparity map. The simple idea of the proposed method is to pick out these approximately constant connected regions. One of the difficulties is that the disparity values of an obstacle surface are only constant in the ideal case. In practice, the depth of obstacles varies with their shape. Therefore, a certain amount of disparity variation needs to be tolerated. Of course, the disparity tolerance needs to take into account the decrease of the distance resolution for small disparity values.

The tolerance on the disparity difference within a region $\Delta_{disp}(d)$ can be specified as a function of the maximal depth variation of an obstacle surface Δ_Z. This depth constraints can be translated to a corresponding

limit on the disparity variation within a region of the disparity maps and depends on the current disparity d:

$$\Delta_{disp}(d) = \frac{d^2 \Delta_Z}{\alpha b + d\Delta_Z} \quad (2)$$

Now that the disparity tolerance is specified, a simple flood-fill algorithm is applied: given a starting point p^*, all neighboring pixels p that satisfy the disparity difference constraint $d(p) - d(p^*) < \Delta_{disp}(d(p^*))$ (2) are grouped. A last question to answer is "which pixels should be considered as starting points?". The naive answer would be to apply the flood-fill to a discrete grid of sub-sampled image positions. A better strategy consists in selecting the maximal disparity value of the disparity map as a starting point. The pixel grouped with this seed point are then suppressed and the next maximal value is processed. The algorithm stops when the whole image is processed or when the maximal disparity is zero. This strategies enables to extract obstacle regions by starting with the closest obstacle (maximal disparity).

C. Geometrical selection

Finally, since not all approximately fronto-parallel planar surfaces correspond to relevant obstacles, a geometrical filtering step is applied. Based on the 3D geometrical characteristics of the extracted surfaces (position in the scene, width, height), it is possible to select only a subset of objects of interest. This part of the algorithm is of course application dependent. If the focus is on pedestrian detection the constraints will be stronger than for a general obstacle detection algorithm.

V. Experimental results

This section presents some results on real data in the context of pedestrian detection but the algorithm can be applied to general obstacle detection by relaxing the geometrical constraints on the obstacles. The data of this experiments was collected with the LOVe project.

The baseline of the stereo system is about 40cm and the processed images are 320×240.

The geometrical constraints enforced on the extracted connected components are the following:

- image surface: 50 pixels,
- height: 1m to 2m,
- top: < 2.5m,
- bottom: < 0.4m,
- width over height ratio: < 1.

A first series of results is presented in Fig. 3. Another series of results on various urban scenarios is presented in Fig. 4. These results show that the detection method correctly picks out the obstacles verifying these geometrical specifications and that the detected objects indeed correspond to pedestrians.

A few false alarms appear on road scene objects that do comply with the geometrical constraints but do not correspond to pedestrians. This is the case of the back of some partly occluded cars parked on the road side (see

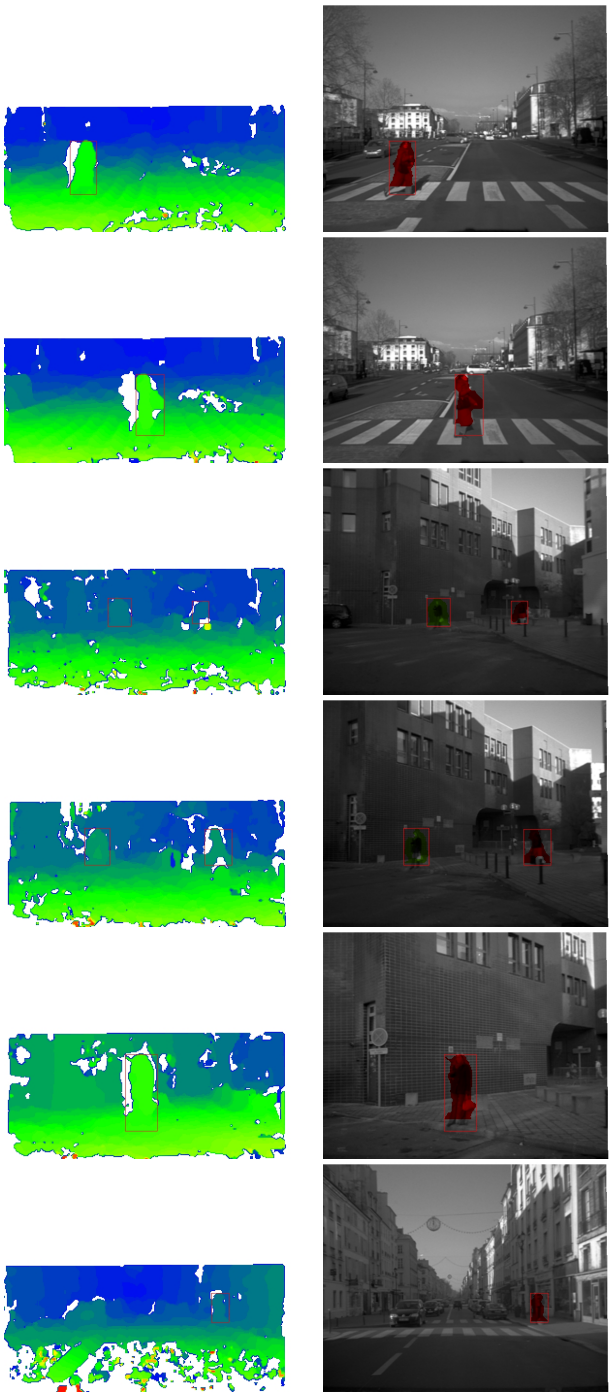


Fig. 3. Disparity maps (right) and corresponding extracted connected components (left). The disparity values decrease from red, over yellow, and green to blue. The red rectangles correspond to the bounding boxes of the detected components. Pixels belonging to a detected component are colored in the gray level images.

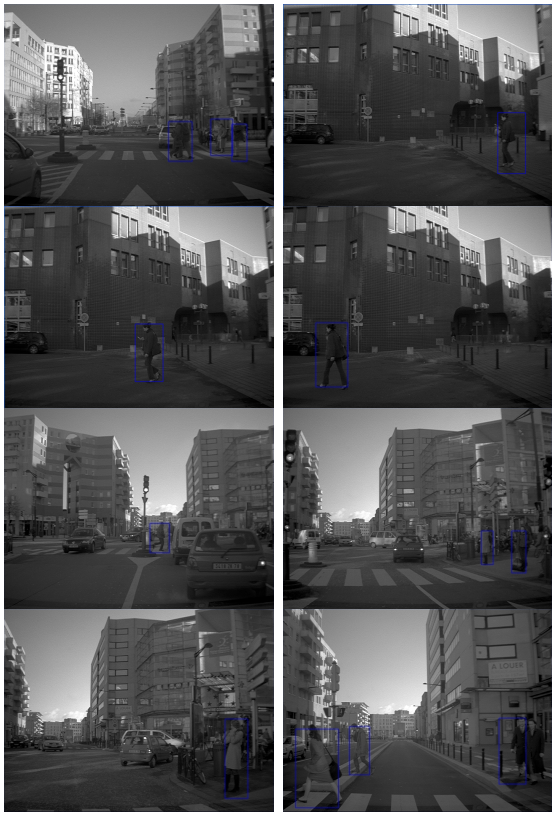


Fig. 4. Detection results for different urban scenes. The blue rectangles correspond to the bounding boxes of the selected connected components.

Fig. 5) and of parcmeters as illustrated on Fig. 3 (last line). A shape recognition step should be able to lift these ambiguities.

Fig. 5. Some false alarms on the back of partly occluded parked cars.

The computation time on a 2Ghz Intel Xeon is about 70ms. The computation of the disparity map corresponds to about 65% of this time. No SIMD optimization were implemented.

Compared to other obstacle segmentation methods the proposed method is simple and intuitive: it does not require any expert knowledge of the segmentation algorithm for tuning the parameters. The detection results are really satisfying since objects verifying the geometric constraints are accurately extracted. All detection related threshold are expressed in explicits units. Most of them are expressed in meters and refer to the size in the real scene.

VI. Conclusion

In this paper, a method for segmenting obstacles from a disparity map was proposed. The method groups neighboring disparity values according to a depth difference constraint. Simple geometrical rules enable to select relevant objects among the extracted components. Effective results were obtained on various real image sequences in an urban environment.

Of course analysing only the disparity map might not be sufficient to obtain a highly robust obstacle detection algorithm. Two aspects might highly improve the robustness : enforcing temporal coherence of successive detections by tracking them and applying shape recognition in order to validate the detected region as relevant obstacles.

VII. Acknowledgment

This work was partially funded by the ANR project LOVe. Thanks to A. Cord for fruitful discussions.

References

[1] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38–50, March 2006.

[2] M. Bertozzi and A. Broggi, "Gold: A parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 62–81, January 1998.

[3] P. Lombardi, M. Zanin, and S. Messelodi, "Unified stereovision for ground, road, and obstacle detection," in *IEEE Intelligent Vehicles Symposium*, 2005.

[4] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection on non flat road geometry through 'v-disparity' representation," in *Proceedings of IEEE Intelligent Vehicle Symposium*, vol. 2, Versailles, France, 2002, pp. 646–651, http://perso.lcpc.fr/tarel.jean-philippe/publis/iv02.html.

[5] Q. Yu, H. Araújo, and H. Wang, "A stereovision method for obstacle detection and tracking in non-flat urban environments," *Autonomous Robots*, vol. 19, no. 2, pp. 141–157, September 2005.

Session II

Multi-sensor perception & navigation

- **Title: Safe and Dependable Operation of a Large Industrial Autonomous Forklift**
  Authors: Ashley Tews

Safe and Dependable Operation of a Large Industrial Autonomous Forklift

Ashley Tews

***Abstract*— For autonomous vehicles to operate in industrial environments, they must demonstrate safe, reliable, predictable, efficient and repeatable performance. To achieve this, two important high level factors are situational awareness and system dependability. The vehicle must be able to identify objects and predict the trajectories of dynamic objects in order to avoid unplanned interaction and to improve performance. In many environments, the vehicle is also required to operate for long periods of time over many days, weeks and months. Towards this goal, the vehicle needs to self-monitor its hardware and software systems, and have redundant primary systems. We have incorporated many of these requirements into our Autonomous Hot Metal Carrier which is a modified 20 tonne forklift used in aluminium smelters for carrying a 10 tonne payload between large sheds, in the presence of other vehicles and people. Our HMC has successfully conducted 100's of hours of autonomous operation in our industrial worksite. The main hardware and software systems will be discussed in this paper with particular focus on the redundant localisation and obstacle avoidance systems. Experiments are described to highlight the performance of the HMC systems in the presence of dynamic objects around a typical worksite.**

I. Introduction

Vehicles operate constantly around industrial worksites. In many applications, they perform repetitive homogeneous tasks such as moving loads from one warehouse location to another. In the aluminium industry, Hot Metal Carriers (HMCs) perform the task of transporting molten aluminium from the smelter (where the aluminium is made) to the casting shed where it is turned into block products. The vehicles weigh approximately 20 tonnes unloaded and resemble forklifts except they have a dedicated hook for manipulating the load rather than fork tines (Figure 1). The molten aluminium is carried in large metal crucibles. The crucibles weigh approximately 2 tonnes and they can hold 8 tonnes of molten aluminium usually superheated above 700 degrees Celcius. Therefore, HMC operations are considered heavy, hot, and repetitive, with safety of operation a significant issue.

Our research is focused towards automating the operations of Hot Metal Carrier-like vehicles. There are many challenges in their operating environment considering they travel inside and outside of buildings. Inside, there is a vast amount of infrastructure, other mobile machines and people. In various areas, there are strong magnetic fields and high temperatures near the molten aluminium vats. Outside, their paths may be surrounded by infrastructure, fences, and their operation may be affected by the environmental conditions: rain, fog, snow, and heat. Research into automating these vehicles and their operations needs to consider the variability in operating conditions to produce repeatable and reliable performance of the task.

Ashley Tews is with the Commonwealth Scientific and Industrial Research Organization, Queensland, Australia Ashley.Tews@csiro.au

Fig. 1. A Hot Metal Carrier in the process of picking up the crucible.

At our worksite, we have fully automated a Hot Metal Carrier and have demonstrated typical operations of a production vehicle. Our vehicle is capable of autonomous start up, shutdown, navigation, obstacle management, and crucible pickup and drop off. It has conducted hundreds of hours of autonomous operations and demonstrated long periods of high reliability and repeatability. The vehicle also has several safety systems incorporated into it to make its operations as safe as possible. The remainder of this paper outlines our research and results.

II. Modules

To be fully capable of conducting all tasks of a manned vehicle, the autonomous HMC needs to address the issues of safety, reliability and repeatability. We have considered these issues when automating the HMC's hardware and software systems. A block diagram of the major hardware components is shown in Figure 2.

The major modules of the system are separated into high level and vehicle level. The high level modules provide commands for controlling the vehicle based on the requested tasks, vehicle state and observed state of the environment.

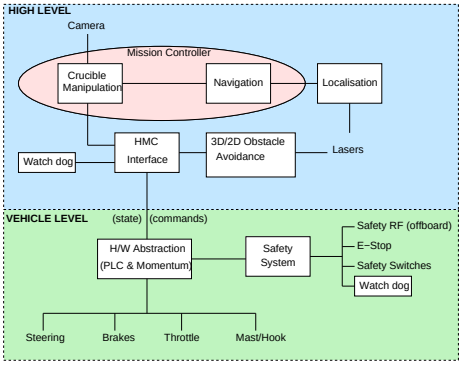


Fig. 2. Overview of the hardware architecture.

The vehicle level modules provide vehicle state information and act as the interface to the vehicle's control systems. They also take care of several low level safety interfaces including physical interlocks, heartbeat monitors between critical systems, and e-stop control.

The vehicle has a light stack on the hood to provide a basic visual indication of its state. The lights indicate whether an e-stop is active, user intervention is required, if the vehicle is operating in autonomous mode, as well as two programmable status indicators that can be used to indicate if a monitored system's parameters exceed or drop below a threshold value (e.g. pneumatic system pressure).

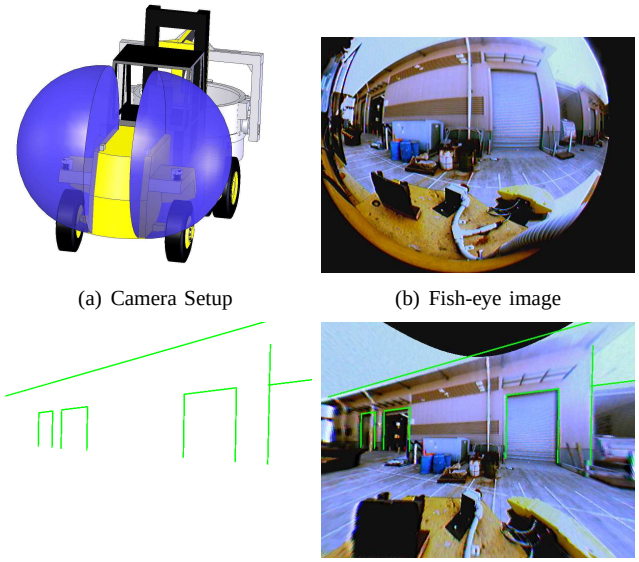
To allow the HMC to conduct autonomous operations safely without the requirement of a safety supervisor to be in the cabin, a RF safety remote is part of the low level interface. This allows the supervisor to be outside of the cabin to monitor operations. The unit has several programmable switches and an e-stop switch to stop the vehicle in an emergency.

The remainder of this section describes the main high level modules.

A. Redundant Localisation

A fundamental requirement for any autonomous vehicle conducting reliable operations is localisation. It forms the basis of any high level navigation, path planning and obstacle avoidance systems. To achieve high reliability, single points of failure need to be reduced or removed completely. Many localisation system use a single type of sensor or fuse sensors into a single system. A hardware or sofware fault with these systems can render the localisation useless. Consequently, the vehicle may have little choice but to signal a fail and wait to be rescued. Using redundant hardware and software systems provides many benefits including the ability to continue with the complete failure of a system as well as the ability to cross-reference systems for bootstrapping, validity checking, and can also be used for offline data fusion.

The HMC's localisation consists of independent vision and laser-based systems. The vision-based localisation system is described in detail in [1]. It conists of a firewire fisheye camera mounted on each of the front mudguards that provide colour images back to an onboard computer (see Figure 3). The images are exposure compensated and edge-features extracted. Edges consist of the outline of major pieces of infrastructure such as sheds and doorways. The resulting edge-map is compared to an *a-priori* map generated offline from surveyed coordinates. The matches are determined probabilistically using a particle filter. The laser-based system uses the four outer lasers on the HMC (see Figure 4) and retro-reflective tape that forms artificial beacons highly visible on the lasers' intensity channel. The beacons have been placed at irregular intervals around the worksite with a maximum separation of 30m. Their locations have been surveyed and recorded in a database that is stored on the HMC. The system compares a sensed beacon constellation with the database to triangulate the vehicle position. A particle filter is also used for this purpose. The accuracy of this system is dependent on the density of sensed beacons and around our site, it is sufficient to allow the large HMC to navigate accurately through narrow doorways and roadways



(a) Camera Setup (b) Fish-eye image

(c) 3D-edge-map of buildings (d) Un-distorted image with projected 3D-edge-map

Fig. 3. Examples of the vision-based localisation system. Two fish-eye cameras are placed at the front of the vehicle facing sideways (a and b). The blue hemispheres represent the field of view of the cameras. A surveyed edge map of the buildings (c) can be tracked in the images (d).

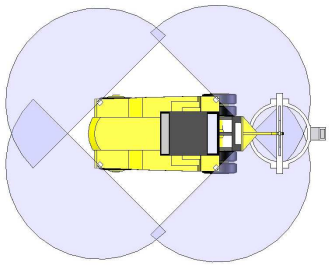


Fig. 4. The HMC's coverage from lasers mounted on the corners.

Fig. 5. Entry to the storage shed where the crucible gets dropped off. Note the clearance between the vehicle and the doorway sides is less than 20cm. The vehicle successfully traverses through the doorway using waypoints which demonstrates the accuracy and repeatability of the localisation, navigation and control systems.

(e.g. Figure 5), some of which have a clearance of 20cm.

The combined localisation system works by use of an arbitration mechanism that compares the output and confidence of the vision and laser localisers. If the primary system has a low confidence or fails, the arbitrator promotes the secondary system to the primary and continues to monitor both for failure and recovery. The localisation output from the arbitrator is sent to the navigation module so any single system failure is transparent to vehicle operations. More details of this system are described in [2].

B. Obstacle Detection

The obstacle detection systems consist of one of the most important safety aspects for any autonomous vehicle. We define an 'obstacle' as a significantly sized object that comes close to, or intersects the vehicle's volumetric trajectory. The volumetric trajectory consists of the bounding volume of the vehicle projected along its planned path. This includes overhangs such as the top of a shed door opening, side obstructions, and objects above a certain size on the ground. It is very difficult or expensive to outfit a vehicle such that it is entirely shrouded by a protective sensor curtain that can detect any object approaching or too close to the vehicle. As a result, the HMC uses 2D and 3D obstacle detection systems. These are supplementary systems that run in parallel and affect the vehicle's operations in different ways. These systems are described next.

1) 2D Obstacle Detection: The role of the 2D Obstacle Detection system is to provide a reactive protective envelope around the entire vehicle such that the vehicle will reduce speed and stop as an object approaches. This system is implemented using scanning laser rangefinders located at each corner of the vehicle, mounted approximately 1.4 m from the ground as shown in Figure 4.

These lasers are mounted with a slight downward tilt so they intersect the ground at around 25-30m. This module interacts directly with the hardware interface layer module (HMC Interface) to override any control commands and reduce the vehicle's velocity depending on the range of the object. It has two modes of operation depending on whether the crucible is on or not. When the crucible is on, it is detected in the rear laser scans and consequently, a shaped detection envelope is used instead. In this mode, the vehicle has a blind spot behind the crucible. In typical operations with the crucible on, the vehicle will only reverse when it is dropping off which is less frequent than other operations. However, we are addressing the blind spot issue as part of future work.

A second issue with using planar laser scans is that objects are only detected within the laser plane. Any obstacle above or below the scan is not detected. As a result, the main purpose of this system is to detect people close to the vehicle or nearby infrastructure (e.g. buildings, bollards or parked vehicles). In operation, the vehicle slows when it approaches the obstacle, or the obstacle approaches it until either the object is close enough to warrant the vehicle to halt or it passes. If the object is too close (approximately 50cm), the vehicle will remain stationary until the operator intervenes to remove the object, or drive the HMC around it manually.

2) 3D Obstacle Detection: The 3D obstacle detection system's primary purpose is to provide a more thorough analysis of the path in front of the vehicle. It consists of a system using a laser mounted above the cabin. The laser has a horizontal scan plane that intercepts the ground approximately 25m in front of the vehicle. This allows approximately eight seconds for the vehicle to come to a halt if travelling at high velocities around 3.0 m/s. An obstacle is determined as an object higher than approximately 5cm that lies in the path of the vehicle. The path is determined from the vehicle's current position past the next waypoint. The system works by accumulating scans as the vehicle travels. The ground plane is extracted from these scans and any object projecting from it identified as traversable or not. If it is not, the system sends a signal to the hardware interface to stop the vehicle and signal that an obstacle has been encountered. This signal consists of a flashing light on the vehicle's status light stack and sending a message through the software system. The vehicle remains halted until the object is removed and the status cleared by the operator via the safety remote. Manual, rather than automatic clearing of the status is a safety issue since in general, the vehicle's path should be clear and any unexpected object detected may indicate a problem in that area of the worksite.

C. Mission Controller

The high level mission controller directs the navigation, tasking and path planning components as shown in Figure 6. The Mission Controller is responsible for switching between tasks and monitoring their performance. A task may be "drive along a section of road", "drop off the crucible", "start up the engine" or even "blow the horn". Currently a mission is a sequence of tasks with each task returning its status during execution. Once a task has finished, the Mission Controller selects the next task. Contingencies occurring during task execution cause the Mission Controller to select the contingency sub-task for that task. For example, a missed

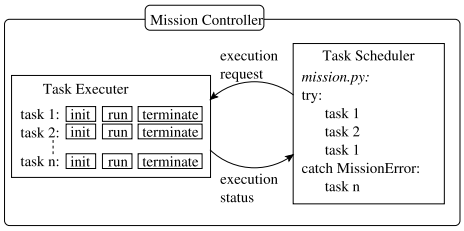


Fig. 6. Overview of the mission control architecture.

Fig. 7. The visual fiducials used to uniquely identify the crucible in the environment.

crucible pick up will trigger a “missed approach” signal and the HMC will move away from the crucible and retry the approach manoeuvre.

The mission controller is a generic component of our system: only the task implementations are specific to the HMC. For this reason, it is currently used on several of our platforms, including an autonomous submarine [3].

D. Human-Robot Interface

The ultimate goal of an autonomous industrial vehicle is for it to be dependable enough to conduct tasks out of sight of an operator. To allow this, the vehicle needs to have some level of offboard control and the ability to report status and sensor data to a safety supervisor who may be monitoring several vehicles simultaneously.

The most basic level of offboard control consists of a remote e-stop that can be manually or automatically triggered. At more advanced levels, the vehicle may be fully controlled offboard by either a computer or physical interface (manual control panel and joystick), with full sensor displays, allowing immersive tele-operation.

Our system consists of a small remote RF portable control unit that has an e-stop, several programmable function switches and a range of approximately 150m. The unit sends a heartbeat signal out periodically which is received by the onboard RF receiver which is hardwired into the e-stop safety PLC circuitry on the vehicle. If a signal is not received within several milliseconds, an e-stop is initiated on the vehicle. We have programmed the switches to perform the functions of halting the vehicle, sounding the horn, and resetting from a ’detected obstacle’ event. The halt function forces the vehicle to stop moving and freezes all controls. Upon release, the vehicle will continue from that state. This function is particularly useful when testing.

The vehicle also outputs data from its internal sensors (e.g. engine parameters, mast information, brakes etc.) and external sensors (lasers and cameras) for external viewing. Visualisation software allows the safety supervisor to monitor all systems on the vehicle.

E. Object Detection

Object detection in the system consists of detecting the crucible for pickup operations, and offboard detection and classification of dynamic objects in the environment. The pickup system is based on visually recognising the crucible in the environment [4]. Due to the similarity of the crucible’s round profile with other objects in the worksite, such as drums, the crucible is uniquely marked with self-similar landmarks as shown in Figure 7. Cameras are mounted on the mast of the vehicle looking rearwards for crucible detection.

The system has different modes of operation depending on whether the crucible’s location is known or not. If it is, which would be the case if the location was recorded when it was dropped off, the cameras are directed to locate the markers on the handle. Once positively identified, the relative location of the crucible is calculated with respect to the hook on the HMC. The vehicle then visually servos to the pickup point on the crucible where the remainder of the pickup procedure is managed as a task in the mission controller. If the location is known only approximately within a 20 by 20 m area, the system will execute a distributed search plan for the cameras to locate it. Once they have, a normal visual servo ensues. This is known as a ’long range’ pickup.

The offboard system is in its preliminary stages at present and consists of a static webcam monitoring one of the common areas for HMC operations. The purpose of this system is to track and classify objects in the scene to provide the HMC with greater situational awareness and offboard localisation ability. The system is based on [5], with enhancements to the classification part of the system. Basically, the system consists of:

1. Determining the background image
2. Performing background subtraction to highlight moving parts of the image
3. Merging proximally close moving parts into single blobs and tracking the blobs
4. Classifying the blobs as either ’vehicle’ or ’person/group’.

The system is capable of tracking and identifying multiple various dynamic objects in a scene, in sunlight and rain. It can handle objects being temporarily occluded or objects crossing paths. Examples of classification are shown below in Figure 8.

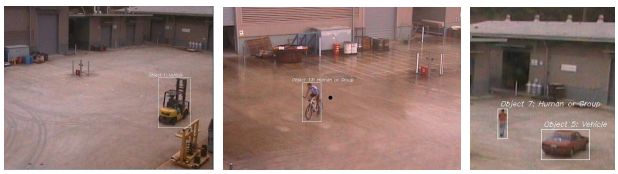


Fig. 8. An example of the various types of dynamic objects tracked. From left to right - forklift, cyclist in the rain, and a person after egressing a car.

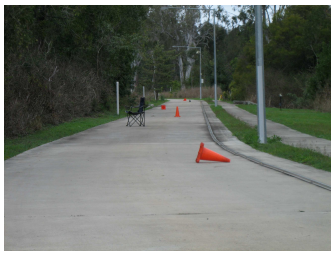


Fig. 9. Traffic cones and chairs used to test the 3D obstacle detection system.

F. Other Modules

There are many other modules that complete the HMC's systems. These include the hardware interface, various safety systems including physical interlocks and heartbeat checks, navigation and crucible manipulation. The operation of the low level interface and safety modules are beyond the scope of this paper. The navigation module is based on waypoint traverses through pre-programmed path segments. The segments are stored in a mission database file and selected as part of the mission script. The inter-operation of the mission controller and navigation system is basic but effective. Picking up and dropping off the crucible are also basic programmed operations that do not vary once the parameters encoding the vehicle position versus the mast motion are tuned.

III. Experiments

A. Obstacle Detection

This consisted of testing the 2D and 3D obstacle detection systems.

2D Obstacle Detection: The 2D tests consist of placing a tall object in the path of the vehicle during forward and reverse manoeuvres, for each corner laser. In each case, the vehicle would slow to a stop as it approached the object. The system was also tested with people walking towards the HMC from various peripheral locations. The HMC perfomed as expected by slowing to a halt as the person approached.

3D Obstacle Detection: The 3D obstacle detection system was tested with a variety of obstacle shapes and sizes along different trajectories of the HMC. Example objects are shown in Figure 9. These objects were placed in the HMC's path during a prolonged experiment. The obstacle detection system correctly determined that each object was an obstacle which would then halt the vehicle when it was within approximately 15 m. The safety supervisor removed the object and reset the 'obstacle detected' system via a switch on the safety remote. The vehicle continued until then next object was found. A screenshot visualising the data on detection of a non-traversable object is shown in Figure 10.

Tests were also conducted with smaller objects consisting of chunks of concrete which were considered traversable by a human operator. In these cases, the vehicle would continue over them.

Fig. 10. A 2D visualisation of an 'obstacle detected' event in the 3D obstacle detection system. The HMC is the yellow object with the grey crucible attached behind it (left). The HMC's path is shown as the black line projected to the right. Environment features are shown in black and pink with the groundplane as the green dots. Along the vehicle's projected path is a red object (traffic cone). Since this object occurs within the width of the vehicle along its path, it is considered an obstacle.

B. Redundant Localisation

The redundant localisation system was tested around our main workarea as shown by the blue square in Figure 11. The area is surrounded by buildings which is well-suited to the vision-based localisation method described previously. The main experiment involved simulating a power failure in the primary laser-based localisation system which reduced its confidence values ([1]). Upon detecting this, the arbitrator switched the primary localisation source to the vision-based localiser and the vehicle continued operations. The laser-beacon localisation system was then brought back online and since it produces slightly higher accuracy and therefore is considered as the primary localisation source, the arbitrator switched back to using its outputs.

C. Long Duration Experiments

Three significant long duration experiments have been undertaken in the project to date. They consist of a two, five, and eight hour trial with the HMC conducting typical operations.

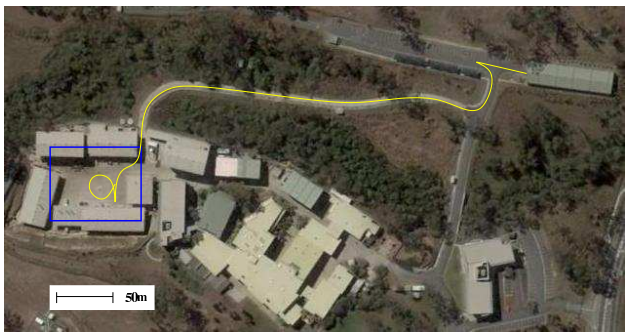


Fig. 11. The path (yellow) of the 2 hour experiment. The crucible pick up and drop off occurred in the open area at the end of the path on the left and the in-shed operations were conducted in the large shed on the upper right. The 5 hour experiment was conducted in the large area surrounded by buildings annotated by the blue box.

Five Hour Trial: the purpose of this trial was to test the integrity of all hardware and software systems continuously operating over five hours. The experiment was conducted in the area indicated by the blue square shown in Figure 11.

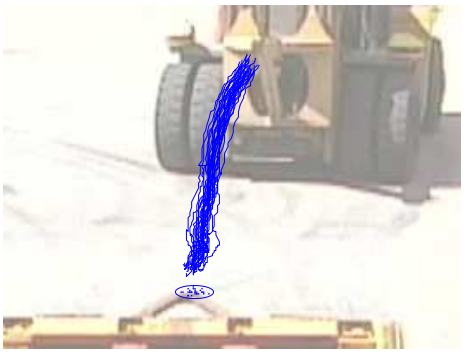


Fig. 12. Transposition of hook path for 29 crucible pickups undertaken at one of the pickup locations during the five hour trial. For reference, the width of the pickup point on the handle of the crucible is approximately 20cm.

The HMC's task was to pick up and drop off the crucible at opposite ends of that area with navigation loops in between tasks. The vision-based crucible detection system was used for locating and servoing to the crucible during pickups. The vehicle undertook the five hour test with the only halt being when the battery on the safety remote had to be replaced. This triggered an e-stop on the vehicle which was then reset and it continued on from where it stopped in the mission. Statistics from this test are shown in Table I. While it is difficult to determine the accuracy of the vehicle and crucible localisation systems due to the lack of a reliable ground truth (GPS is ineffective around built environments mainly due to multi-pathing), upon analysing the log files recorded during the test showed a maximum path spread of 0.3m over all paths with the average being less than 0.2m. The accuracy of the crucible pickups occurring at one end of the test area, which represent the accuracy of the vision-based crucible recognition system is shown in Figure 12.

Two Hour Trial: from the success of the systems tested in the five hour experiment, a trial was conducted with a longer traverse path along a narrow road and a crucible dropoff point inside a shed with a narrow entry and filled with equipment. This main path is shown in yellow in Figure 11 and the shed entry in Figure 5.

Three techniques for locating the crucible were tested. Two were vision-based as described in Section II-E. The third was based on servoing to the dropoff location of the crucible recorded from the laser-beacon localisation system. This provided a test of the accuracy of the laser-beacon localiser since any error in location would result in the HMC trying to pick the crucible up from the wrong location.

The mission script required the HMC to autonomously start up, traverse to the crucible scan location and conduct a 'long range' visual pickup. It would then traverse to the storage shed, drop the crucible off inside, drive out and conduct a laser-beacon localiser pickup in the shed. It would then traverse back to the start location, drop the crucible off, navigate around the area to a point where it would conduct a normal vision-based crucible pickup. This cycle to and from the storage shed constituted the remainder of the mission until the last cycle where the crucible was placed in its 'home' location and the HMC parked in its shed and shut down. All phases of the trial were conducted successfully. More details about the five and two hour experiments can be found in [4].

TABLE I

KEY STATISTICS FROM THE 5 AND 2 HOUR EXPERIMENTS

Experiment	Total Dist.	Cycle Dist.	Velocity Range	Cruc. Ops.
5 hr	8.5 km	0.3 km	-1.1 : 1.6 m/s	58
2 hr	6.5 km	0.93 km	-1.4 : 3.0 m/s	14

Eight Hour Trial: The purpose of this trial was to test automated door control, 3D obstacle detection and vehicle scheduling over a shift of normal vehicle operations. The mission was written such that every hour, the vehicle would signal an operator to enable the physical safety interlocks to allow it to conduct a task sequence. The sequence consisted of starting up in its shed and requesting the shed door to open via wireless communication to a receiver on the door built specifically for the purpose. Once the door signalled it was open, the HMC would move out, request the door to close and conduct the crucible pickup - navigation - dropoff cycle described in the two hour trial. Upon completing the 20 minute cycle, it would request the door to open, drive in and park with a final request to close the door. All operations were conducted successfully during the eight hours.

IV. DISCUSSION

It is important for autonomous vehicles operating in environments with large amounts of infrastructure and in the presence of dynamic objects to be able to conduct repeatable, safe, predictable and reliable operations. Dynamic objects can manifest as people or vehicles moving about the environment, sometimes within close proximity to the robot. To provide the required dependable operations, the vehicle should have redundant self-monitoring systems that are fault-tolerant and where possible have redundant backups. Outside the vehicle, it needs to be 'situationally aware' of its surroundings with respect to its task. Local observations taken from environment sensors such as lasers may be insufficient to determine potential collisions with unseen dynamic objects. Offboard systems such as webcams mounted to infrastructure, or even the perception from other mobile bases can be used to augment this extra sensing.

We are in the process of providing these functionalities with the autonomous Hot Metal Carrier project. Many of the systems described in this paper have been designed to accommodate these requirements. In particular, the localisation, obstacle detection and object recognition systems. While the object recognition system is currently offboard the vehicle, it is capable of tracking and localising dynamic objects to report back to the HMC. We are currently undertaking experiments to demonstrate this utility. While the HMC consists of several basic systems, it has been successfully

conducting autonomous operations over hundreds of hours of demonstrations and tests. The fundamental systems have proven reliable, but need to facilitate the redundancy and situational awareness capbilities mentioned above.

Acknowledgements

This work was funded in part by CSIRO's Light Metals Flagship project and by the CSIRO ICT Centre's Dependable Field Robotics projects. The author gratefully acknowledge the key contributions of the Hot Metal Carrier project team; in particular: Cedric Pradalier, Robert Zlot, Polly Alexander, Paul Flick, and Jonathan Roberts. The author would also like to thank the Engineering Support team for their maintenance efforts.

REFERENCES

[1] S. Nuske, J. Roberts, and G. Wyeth, "Outdoor visual localisation in industrial building environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, U.S.A, May 2008.

[2] J. Roberts, A. Tews, and S. Nuske, "Redundant sensing for localisation in outdoor industrial environments," in *Proceedings of the 6th IARP/IEEE-RAS/EURON Workshop on Technical Challenges for Dependable Robots in Human Environments*, 2008.

[3] A. Negre, C. Pradalier, and M. Dunbabbin, "Robust vision-based underwater target identification & homing using self-similar landmarks," in *Proceedings of International Conference on Field and Service Robotics*, Chamony, France, 2007.

[4] C. Pradalier, A. Tews, and J. Roberts, "Vision-based operations of a large industrial vehicle: Autonomous hot metal carrier," *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 243–267, April-May 2008.

[5] O. Javed and M. Shah, *Computer Vision - ECCV 2002*, ser. Lecture Notes in Computer Science. Spinger Berlin/Heidelberg, 2002, vol. 2353/2002, ch. Tracking And Object Classification For Automated Surveillance, pp. 439–443.

Session III

SLAM, Localization, Reconstruction

- **Title: Detection Likelihoods for Safer Occupancy Mapping** *(invited paper)*
  Authors: John Mullane, Martin Adams, Wijerupage Sardha Wijesoma

- **Title: Experimental Comparison of Bayesian Outdoor Vehicle Localization Filters**
  Authors: Alexandre N. Ndjeng, Dominique Gruyer, Alain Lambert, Sébastien Glaser, Benjamin Mourllion

- **Title: Predictive Lane Detection for Simultaneous Road Geometry Estimation and Vehicle Localization**
  Authors: Chenhao Wang, Zhencheng Hu, Tomoki Maeda, Naoko Hamada, and Keiichi Uchimura

- **Title: Cognitive Localization of 3D Objects Symbolically Given Navigational Cues** *(invited paper)*
  Authors: Sukhan Lee, Hyunjun Kim, Zhaojin Lu, and Harry Hung

- **Title: Laser scanner based SLAM in real road and traffic environment**
  Authors: Olivier Garcia-Favrot and Michel Parent

Session III

SLAM, Localization, Reconstruction

- **Title: Detection Likelihoods for Safer Occupancy Mapping** *(invited paper)*
  Authors: John Mullane, Martin Adams, Wijerupage Sardha Wijesoma

Detection Likelihoods for Safer Occupancy Mapping

John Mullane
Martin Adams
Wijerupage Sardha Wijesoma

School of Electrical & Electronic Engineering
Nanyang Technological University
Singapore

Presentation Outline

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Presentation Outline

1. Introduction
   - Motivation
   - Environment Representation
   - Stochastic Estimation
2. The GBRM Framework
   - The Range-based Recursion
   - The Detection-based Recursion
   - Verification: Ideal Likelihoods
3. Case Study: MMW Radar Map Estimation
   - The Measurement Likelihoods
   - Filter Implementations
   - Filter Analysis
4. Conclusions & Future Directions
   - Conclusions & Future Directions

Motivation

- Accurate map estimation is critical for safe and reliable autonomous navigation

Motivation

- Accurate map estimation is critical for safe and reliable autonomous navigation
- Exteroceptive sensors can be noisy, stochastic methods popular

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Motivation

- Accurate map estimation is critical for safe and reliable autonomous navigation
- Exteroceptive sensors can be noisy, stochastic methods popular
- As sensing noise increases, performance of current occupancy grid approaches deteriorate

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Motivation

- Accurate map estimation is critical for safe and reliable autonomous navigation
- Exteroceptive sensors can be noisy, stochastic methods popular
- As sensing noise increases, performance of current occupancy grid approaches deteriorate
- ✔ Examine mathematical foundation of standard occupancy measurement likelihoods

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Motivation

- Accurate map estimation is critical for safe and reliable autonomous navigation
- Exteroceptive sensors can be noisy, stochastic methods popular
- As sensing noise increases, performance of current occupancy grid approaches deteriorate
- ✔ Examine mathematical foundation of standard occupancy measurement likelihoods
- ✔ Improve accuracy of maps estimated, as sensor noise increases

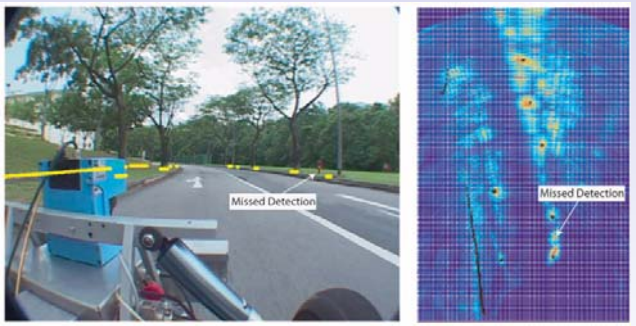
Environmental Perception

- ✔ Provides real-time situational awareness
- ✔ Provides absolute correction data for real-time path estimation

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Environmental Perception

- ✔ Provides real-time situational awareness
- ✔ Provides absolute correction data for real-time path estimation



- Measurement Uncertainty:
  - ✘ Measurement noise
  - ✘ Spurious measurements
  - ✘ Detection uncertainty
  - ✘ Data association uncertainty

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Presentation Outline

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Sample Environment

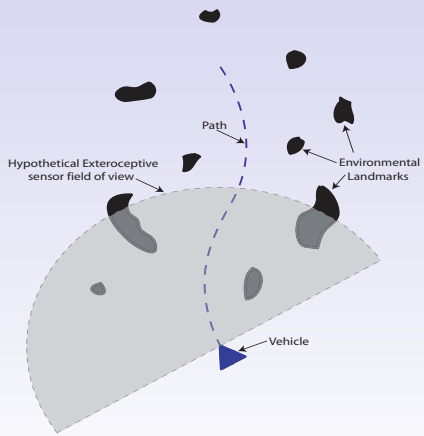


Figure: *A General 2D Autonomous Navigation Scenario.*

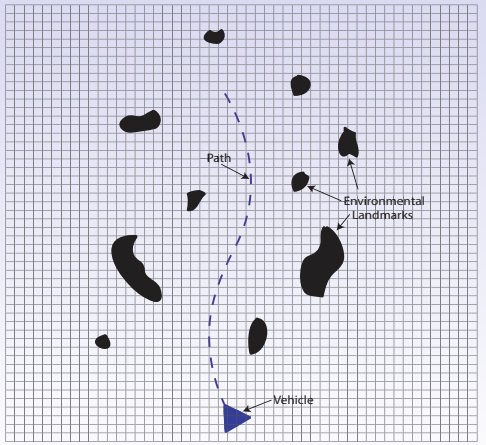
Map Representation

- ✔ Represent the map, M, as a mathematical object.

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Map Representation

- ✔ Represent the map, M, as a mathematical object.



Grid-based Map

- [Moravec, '85]
- [Elfes, '89]
- [Martin, '96]
- [Konolgie, '97]
- [Pagac, '98]
- [Gutmann, '99]
- [Foessel, '02]
- [Thrun, '02]
- [Hahnel, '03]
- [Grisetti, '03]
- [Thrun, '03]
- [Yang, '06]

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Presentation Outline

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Dealing with Measurement Uncertainty

- ✔ Environmental Estimation: Robotic Mapping
- ✔ Bayesian approach, widely accepted in robotics
- Assuming vehicle path is known (RM):

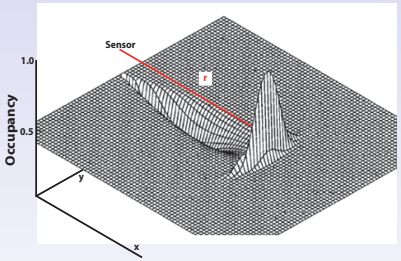
$$p_{k|k}(M_k|Z^k, X^k, u^{k-1}, X_0) = \frac{g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)}{\int g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, u^{k-1}, X_0)dM_k}$$

$p_{k|k}(M_k|Z^k, X^k, u^{k-1}, X_0)$: encapsulates all uncertainty about the map at time k.

The Measurement Likelihoods

- Widely adopted in the GBRM literature.
  - **Grid Maps**:

$$G_k(Z_k = r | M_k = \text{GRID}, X_k)$$ [Elfes, '89]

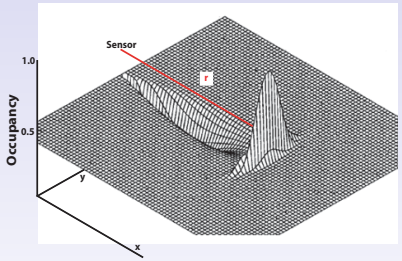


Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

The Measurement Likelihoods

- Widely adopted in the GBRM literature.
  - **Grid Maps**:

$$G_k(Z_k = r | M_k = \text{GRID}, X_k)$$ [Elfes, '89]



- How are the measurement likelihoods calculated ?

$$G_k(Z_k = r | M_k = E, X_k) \textbf{ ?}$$

$$G_k(Z_k = r | M_k = O, X_k) \textbf{ ?}$$

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Motivation Summary

- Safe autonomous navigation requires accurate map estimates

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

Motivation Summary

- Safe autonomous navigation requires accurate map estimates
- ✘ In challenging environments (landmarks of various shapes and sizes) and noisy sensors (radar / sonar), incorporation of uncertainty in to filter recursion is critical

Motivation Summary

- Safe autonomous navigation requires accurate map estimates
- ✘ In challenging environments (landmarks of various shapes and sizes) and noisy sensors (radar / sonar), incorporation of uncertainty in to filter recursion is critical
- ✘ Occupancy mapping likelihoods appear to have some inconsistencies

Motivation Summary

- Safe autonomous navigation requires accurate map estimates
- ✘ In challenging environments (landmarks of various shapes and sizes) and noisy sensors (radar / sonar), incorporation of uncertainty in to filter recursion is critical
- ✘ Occupancy mapping likelihoods appear to have some inconsistencies
- ✔ Change the measurement space from range/bearing to detection/non-detection

Motivation Summary

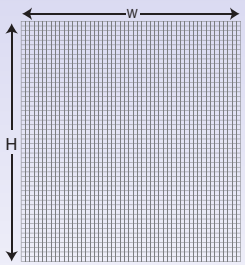
- Safe autonomous navigation requires accurate map estimates
- ✘ In challenging environments (landmarks of various shapes and sizes) and noisy sensors (radar / sonar), incorporation of uncertainty in to filter recursion is critical
- ✘ Occupancy mapping likelihoods appear to have some inconsistencies
- ✔ Change the measurement space from range/bearing to detection/non-detection
- ✔ Improve robustness of occupancy grid framework to noisy environments and sensors

Presentation Outline

Grid Mapping Example

The GBRM Problem

$$p_{k|k}(M_k = [m_1, \ldots, m_{W\times H}]|Z^k, X^k, u^{k-1}, X_0) = \frac{g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)}{\int g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, u^{k-1}, X_0)dM_k}$$



- Bayesian recursive approach
- Measurement uncertainty
- Map occupancy uncertainty
- Decompose map into $W \times H$ independent estimation problems

$$p_{k|k}(M_k|Z^k, X_k) = \prod_{i=1}^{i=W\times H} p_{k|k}(m_k^i|Z^k, X^k)$$

GBRM requires the propagation of the map occupancy state.

Current Approach: The Range-based Recursion

The Range-based GBRM Filter

$$p_{k|k}(M_k = [m_1, \ldots, m_{W\times H}]|Z^k, X^k, u^{k-1}, X_0) = \frac{g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)}{\int g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, u^{k-1}, X_0)dM_k}$$

- State is binary: $M = [O, E]$
- Prediction: $p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)$
- Measurement: Z_k =range/bearing
- Form likelihood: $g_k(Z_k|M_k, X_k)$
- Bayesian Update: $p_{k|k}(M_k|Z^k, X^k, u^{k-1}, X_0)$

Current Approach: The Range-based Recursion

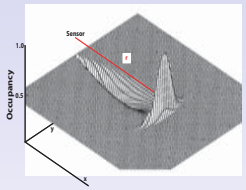
The Range-based GBRM Filter

$$p_{k|k}(M_k = [m_1, \ldots, m_{W\times H}]|Z^k, X^k, u^{k-1}, X_0) = \frac{g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)}{\int g_k(Z_k|M_k, X_k)p_{k|k-1}(M_k|Z^{k-1}, u^{k-1}, X_0)dM_k}$$

- State is binary: $M = [O, E]$
- Prediction: $p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)$
- Measurement: Z_k =range/bearing
- Form likelihood: $g_k(Z_k|M_k, X_k)$
- Bayesian Update: $p_{k|k}(M_k|Z^k, X^k, u^{k-1}, X_0)$

The Measurement Likelihood: State Dependency

$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$

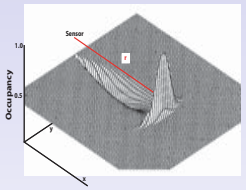


- The likelihood of a range measurement conditioned on the occupancy state and vehicle pose

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

The Measurement Likelihood: State Dependency

$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$



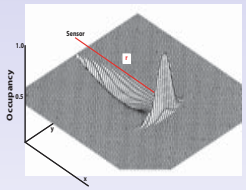
- The likelihood of a range measurement conditioned on the occupancy state and vehicle pose

Q. What is the function that relates $m_{k,(x)}$ and X_k to z_k, where z_k is range reading?

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

The Measurement Likelihood: State Dependency

$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$



- The likelihood of a range measurement conditioned on the occupancy state and vehicle pose

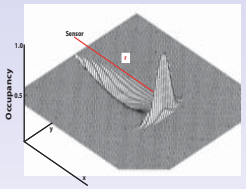
Q. What is the function that relates $m_{k,(x)}$ and X_k to z_k, where z_k is range reading?

A. Use $z_k = \{\text{Detection,No Detection}\}$ to get state dependant measurement equation.

Introduction The GBRM Framework Case Study: MMW Radar Map Estimation Conclusions & Future Directions

The Measurement Likelihood: Uncertainty

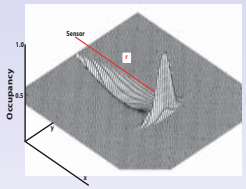
$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$



- Dealing with detection uncertainty and spurious measurements.

The Measurement Likelihood: Uncertainty

$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$

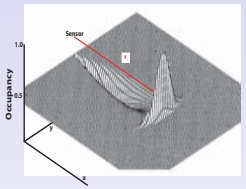


- Dealing with detection uncertainty and spurious measurements.

Q. For no range reading, how is $G_k(z_k = r | m_{k,(x)} = [O, E], X_k)$ defined ?

The Measurement Likelihood: Uncertainty

$$G_k(z_k = r | m_{k,(x)} = O, X_k)$$



- Dealing with detection uncertainty and spurious measurements.

Q. For no range reading, how is $G_k(z_k = r | m_{k,(x)} = [O, E], X_k)$ defined ?

A. Use $z_k = \{\text{Detection,No Detection}\}$ to have a well-defined likelihood.

Current approach: Drawbacks

- Grid-based Framework
  - ✔ Estimation state space: Occupancy
  - ✔ Map representation
  - Measurement Likelihood:
    - ✔ Measurement Noise
    - ✗ State dependent
    - ✗ Detection Uncertainty
    - ✗ Spurious Measurements

Presentation Outline

Proposed Approach: Advantages

- Grid-based Framework
  - ✔ Estimation state space: Occupancy
  - ✔ Map representation
  - Measurement Likelihood:
    - ✔ Measurement Noise
    - ✔ State dependent
    - ✔ Detection Uncertainty
    - ✔ Spurious Measurements

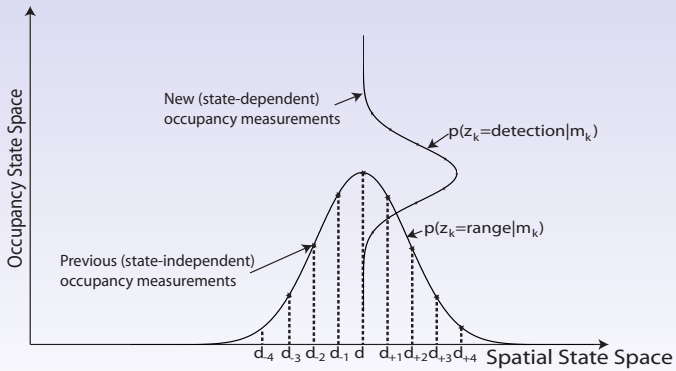
Proposed Approach: The Detection-based Recursion

The Detection-based GBRM Filter

$$p_{k|k}(M_k = [m_1, \ldots, m_{W\times H}] | Z^k, X^k, u^{k-1}, X_0) = \frac{g_k(Z_k|M_k, X_k) p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)}{\int g_k(Z_k|M_k, X_k) p_{k|k-1}(M_k|Z^{k-1}, u^{k-1}, X_0) dM_k}$$

- State is binary: $M = [O, E]$
- Prediction: $p_{k|k-1}(M_k|Z^{k-1}, X^k, u^{k-1}, X_0)$
- Measurement: Z_k =Detection / Non-detection
- Form likelihood: $g_k(Z_k|M_k, X_k)$
- Bayesian Update: $p_{k|k}(M_k|Z^k, X^k, u^{k-1}, X_0)$

Proposed Approach: The Filtering State-Space



Proposed Approach: Key Observations

With z={Detection, Non-Detection}:

the measurement likelihood is state-dependant

the measurement likelihood always exists

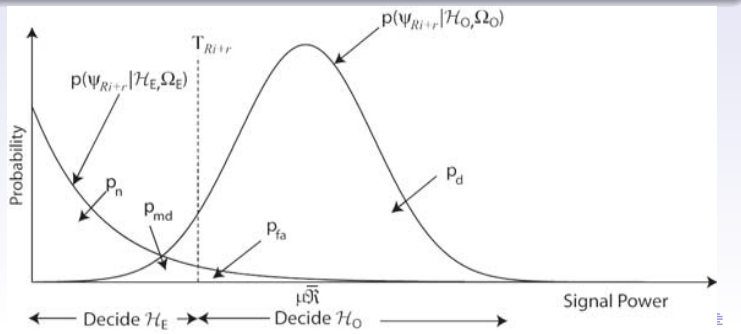
Proposed Approach: Key Observations

With z={Detection, Non-Detection}:

the measurement likelihood is state-dependant

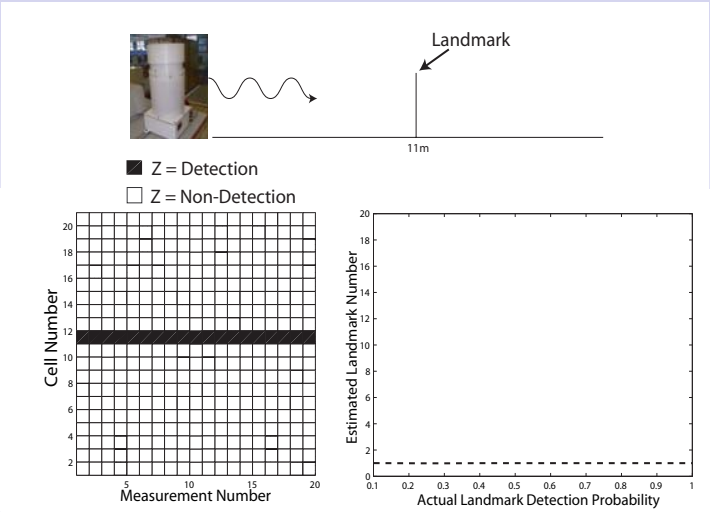
the measurement likelihood always exists

the measurement likelihood becomes the detection statistics



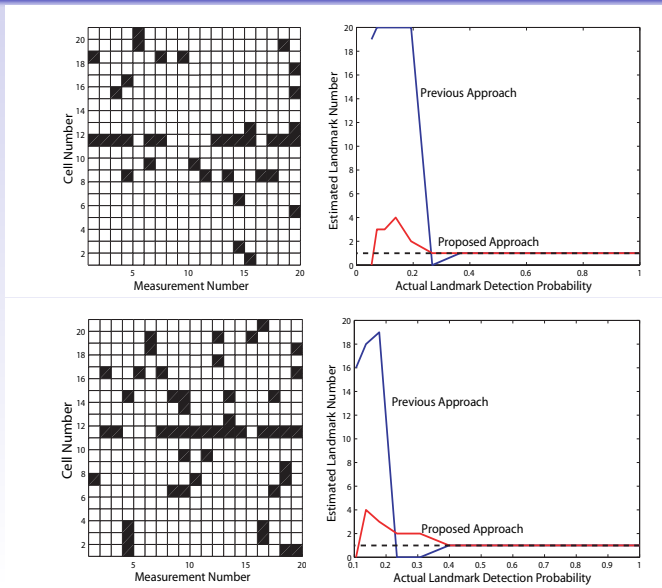
Presentation Outline

Simulation: Known Likelihoods

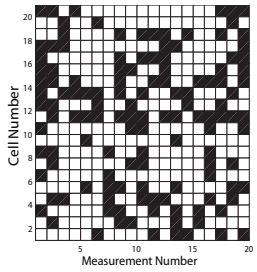


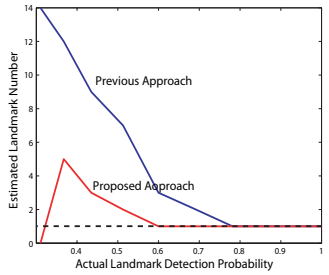
1D Scenario

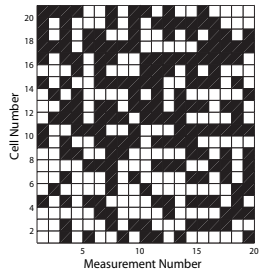
Simulation: Known Likelihoods

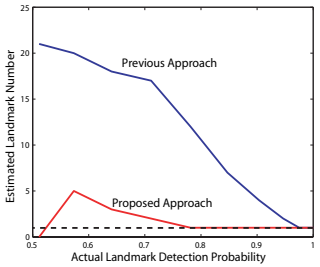


Simulation: Known Likelihoods









Filter Implementation ?

- ✔ Likelihoods are landmark dependent
  - Landmark properties affect its fluctuation model

Filter Implementation ?

- ✔ Likelihoods are landmark dependent
  - Landmark properties affect its fluctuation model
- ✔ Likelihoods are detector dependent
  - Statistical detectors/parameters alter likelihoods

Filter Implementation ?

- ✔ Likelihoods are landmark dependent
  - Landmark properties affect its fluctuation model
- ✔ Likelihoods are detector dependent
  - Statistical detectors/parameters alter likelihoods
- ✔ Likelihoods are sensor dependent
  - Detection theory may differ between sensor - MMWR, LMS, Camera, Sonar etc. etc.

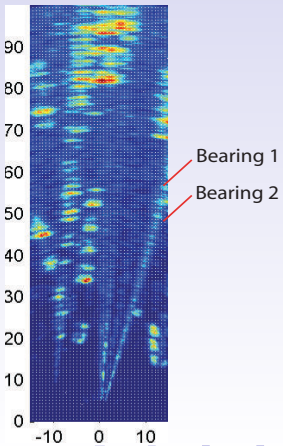
Presentation Outline

The MMW Radar

- Operates at 77GHz
- Returns unprocessed data allowing for custom detector design

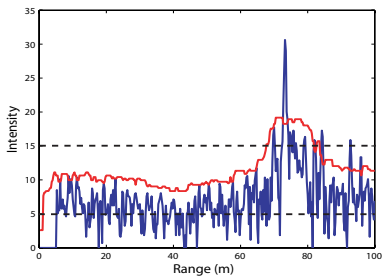
The Detection Problem: A Stochastic Approach

- ✘ Rarely considered in current navigation algorithms
- ✔ Stochastic detectors exploit statistics of underlying signals

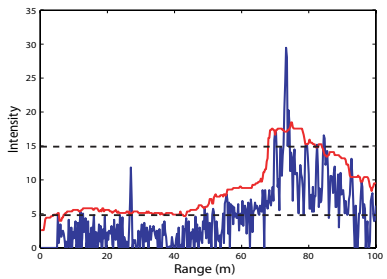


The Detection Problem: A Stochastic Approach

- ✔ Outperform classically adopted constant thresholds
- ✔ Detections (and non-detections) are statistically significant

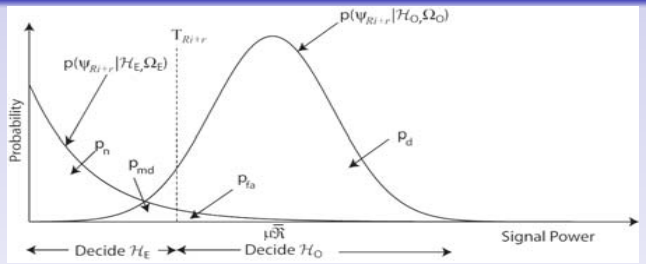


Spectrum at Bearing Angle 1

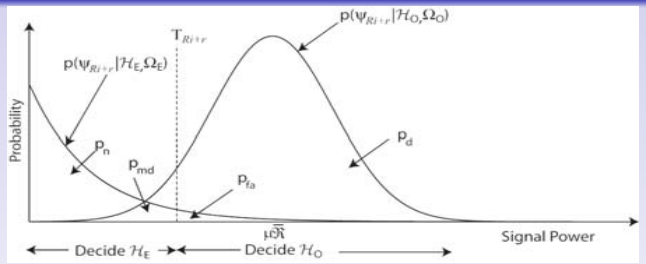


Spectrum at Bearing Angle 2

Detection Statistics



Detection Statistics



$$P_d = \int_0^\infty P[\psi_{Ri+r} \geq T_{Ri+r}|\mathcal{H}_O] f_\mu(\mu) d\mu$$

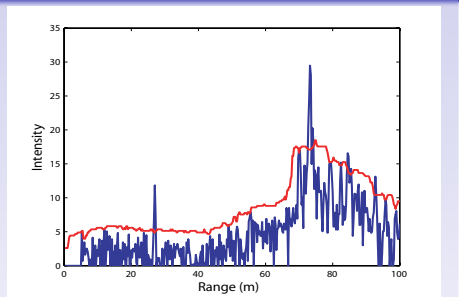
$$P_n = \int_0^\infty P[\psi_{Ri+r} < T_{Ri+r}|\mathcal{H}_E] f_\mu(\mu) d\mu$$

$$P_{md} = \int_0^\infty P[\psi_{Ri+r} < T_{Ri+r}|\mathcal{H}_O] f_\mu(\mu) d\mu$$

$$P_{fa} = \int_0^\infty P[\psi_{Ri+r} \geq T_{Ri+r}|\mathcal{H}_E] f_\mu(\mu) d\mu$$

MMWR: Stochastic Detection

- Form a threshold: $T = \tau\hat{\Omega}_E$
- Popular approaches: OS, CA, GO, SO, ...



- At a given range, r, assuming $\Omega_E = \mu$ then,

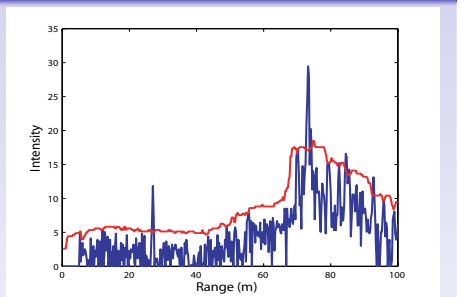
$$g_k(z_k = D|m_{k,(r)} = O, X_k) = \int_0^\infty P[\psi_r \geq T_r|\mathcal{H}_O] f_\mu(\hat{\mu}) d\hat{\mu}$$

$$g_k(z_k = D|m_{k,(r)} = E, X_k) = \int_0^\infty P[\psi_r \geq T_r|\mathcal{H}_E] f_\mu(\hat{\mu}) d\hat{\mu}$$

MMWR: OS-CFAR Likelihoods

- Assume $v(\Omega_E)$ is IID,

$$g_k(z_k = D|m_{k,(r)} = E, X_k) = K$$



- Analgous to data association threshold
  - ✔ χ^2 test accepts a *correct* assignment with a fixed probability
  - ✔ CFAR test accepts an *incorrect* assignment with a fixed probability
  - ✔ Both threshold give *no* information of the converse

MMWR: OS-CFAR Likelihoods

$$g_k(z_k = D | m_{k,(r)} = O, X_k) = \left(1 + \frac{T_r}{1+\bar{\Re}_r}\right)^{-2W}$$

where,

$$T_r = \tau \hat{\mu}_r$$

$$\tau = \arg \min_{\tau} \left(k_{os} \binom{2W}{k_{os}} \frac{(k_{os}-1)!(\tau+2W-k_{os})!}{(\tau+2W)!} - P_{fa} \right)$$

$$\hat{\mu}_r = \Psi_{os,k_{os}}$$

$$\Psi_{os} = \text{sort}([\psi_{r-G-W}, \ldots, \psi_{r-G}] \cup [\psi_{r+G+1}, \ldots, \psi_{r+G+W}])$$

$$\bar{\Re}_r = \frac{\psi_r - \hat{\mu}_r}{\hat{\mu}_r}$$

MMWR: Evaluating the Likelihoods

$$p_{k|k}(m_{k,(x)} = O | z^k, X^k) = \frac{g_k(z_k | m_{k,(x)} = O, X_k) p_{k|k-1}(m_{k,(x)} = O | z^{k-1}, X^k)}{p_{k|k}(z_k | m_{k,(x)}, X^k)}$$

Likelihood	Filter
Discrete probabilistic	Binary Bayes Filter
Discrete evidential	Dempster-Shafer Evidential Filter
Continuous probabilistic	if Gaussian - Kalman Filter if non-Gaussian - Particle Filter.

Presentation Outline

MMWR: Discrete Probabilistic GBRM Filter

$$P_{k|k}(M_k = O | z_{k,(r)} = D) = \frac{Z_{P_d} P_{k|k-1}(M_k = O | Z_{k-1})}{Z_{P_d} P_{k|k-1}(M_k = O | Z_{k-1}) + Z_{P_{fa}} P_{k|k-1}(M_k = E | Z_{k-1})}$$

$$P_{k|k}(M_k = O | z_{k,(r)} = \bar{D}) = \frac{Z_{P_{md}} P_{k|k-1}(M_k = O | Z_{k-1})}{Z_{P_{md}} P_{k|k-1}(M_k = O | Z_{k-1}) + Z_{P_n} P_{k|k-1}(M_k = E | Z_{k-1})}$$

MMWR: Discrete Evidential GBRM Filter

$$\mathfrak{m}_m(C) = \frac{\sum_{A\cap B=C} \mathfrak{m}_z(A)\mathfrak{m}_m(B)}{1 - \sum_{A\cap B=\emptyset} \mathfrak{m}_z(A)\mathfrak{m}_m(B)}$$

$$\mathfrak{m}_z(m_k^F|z_k = \bar{D}) = \frac{Z_{P_{md}}}{Z_{P_{md}} + Z_{P_n} + Z_{P_u}}$$
$$\mathfrak{m}_z(m_k^E|z_k = \bar{D}) = \frac{Z_{P_n}}{Z_{P_{md}} + Z_{P_n} + Z_{P_u}}$$
$$\mathfrak{m}_z(m_k^U|z_k = \bar{D}) = \frac{Z_{P_u}}{Z_{P_{md}} + Z_{P_n} + Z_{P_u}}$$

$$\mathfrak{m}_z(m_k^\emptyset|z_k) = 0$$

MMWR: Continuous Probabilistic GBRM Filter

$$p_{k-1|k-1}(o_{k-1,(r)}|z_{(r)}^{k-1}) \approx \sum_{i=1}^{N} w_{k-1,(r)}^{(i)} \delta_{o_{k-1,(r)}^{(i)}}(o_{k-1,(r)})$$

where,

$$o_{t-1,(r)} = \begin{bmatrix} m_{k-1,(r)} \\ \lambda_{k-1,(r)} \end{bmatrix}$$

$$o_{k,(r)}^{(i)} \sim q(o_{k,(r)}|o_{k-1,(r)}^{(i)}, z_{k,(r)})$$

$$w_{k,(r)}^{(i)} = w_{k-1,(r)}^{(i)} \frac{p(z_{k,(r)}|o_{k,(r)}^{(i)}) p(o_{k,(r)}^{(i)}|o_{k-1,(r)}^{(i)})}{q(o_{k,(r)}^{(i)}|o_{k-1,(r)}^{(i)}, z_{k,(r)})}$$

MMWR: Continuous Probabilistic GBRM Filter

$$p(m_{k-1,(r)}|z_{(r)}^{k-1}) \sim p(m_{k-1,(r)} = 1|z_{(r)}^{k-1}, \Pi)$$
$$p_{k|k-1}(\lambda_{k,(r)}|z_{(r)}^{k-1}) = p_{k-1|k-1}(\lambda_{k-1,(r)}|z_{(r)}^{k-1})$$

$$q(o_{k,(r)}^{(i)}|o_{k-1,(r)}^{(i)}, z_{k,(r)}) = p(o_{k,(r)}^{(i)}|o_{k-1,(r)}^{(i)}).$$

$$p(z_{k,(r)}|o_{k,(r)}^{(i)}) = \frac{p(\psi_r|m_{(r)} = 1, \Omega_O)}{p(\psi_r|m_{(r)} = 0, \Omega_E)}$$

$$\hat{o}_{k,(r)} = \sum_{i=1}^{N} w_{k,(r)}^{(i)} o_{k,(r)}^{(i)}.$$

Presentation Outline

Testing Environment

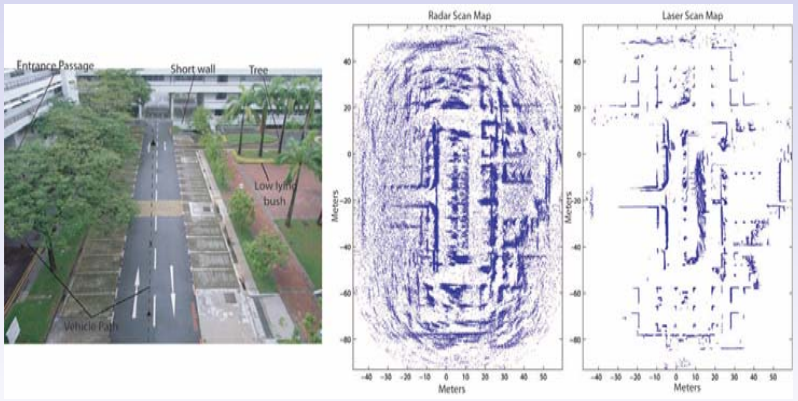


Figure: *Testing Ground overview with corresponding scan maps*

Testing Environment

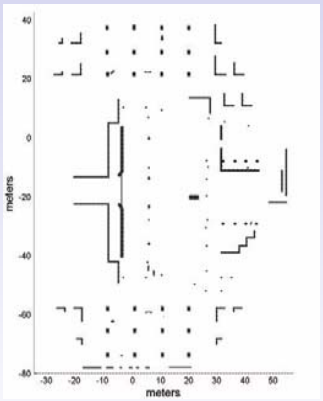


Figure: *Carpark binary ground-truth GB map.*

GBRM: Error Quantification

- Vector Map Comparison
- Sum of Squared Error (SSE) common [Martin, '96], [Collins '98], [Rachlin, '05]

$$\sum_{i}^{q}(m_i - \hat{m}_i)^2$$

✗ Not applicable to outdoor environments

$$NASSE = 0.5\left(\frac{1}{q_O}\sum_{i=0}^{q_O}\left(P(m_k^i|z^{i,k}, m^i=1)-1\right)^2 + \frac{1}{(q-q_O)}\sum_{i=q_O+1}^{q}\left(P(m_k^i|z^{i,k}, m^i=0)-0\right)^2\right)$$

GBRM: Error Quantification

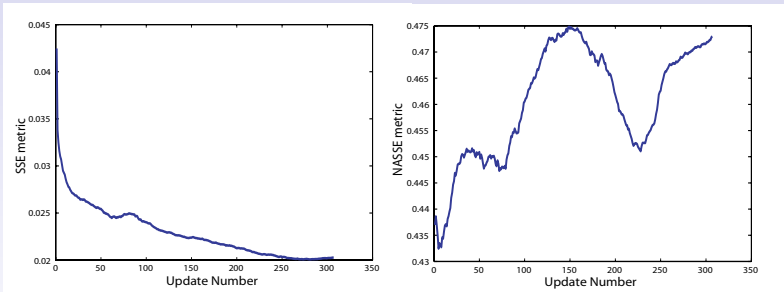


Figure: *Grid-based error metric comparison with localisation error.*

Discrete Probabilistic Implementation

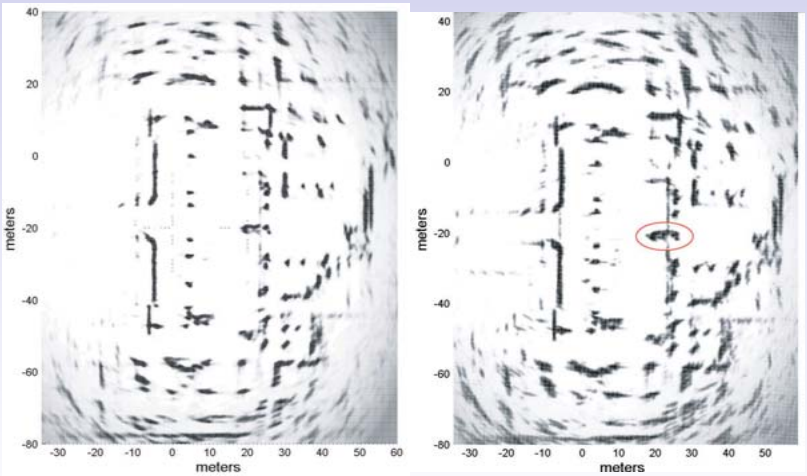


Figure: *Discrete probabilistic detection filter (left) and discrete range filter (right).*

Discrete Probabilistic Implementation

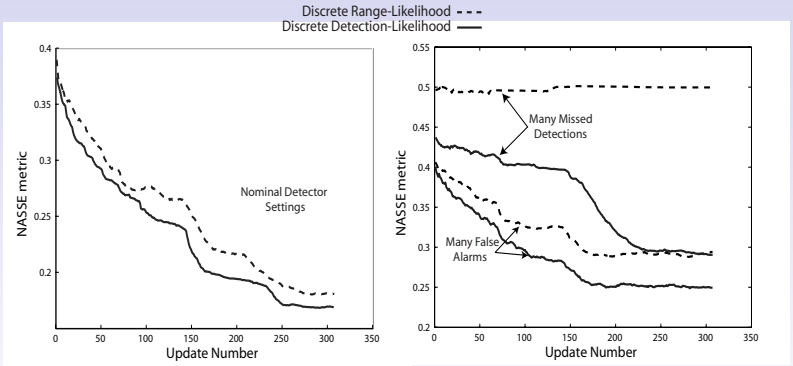
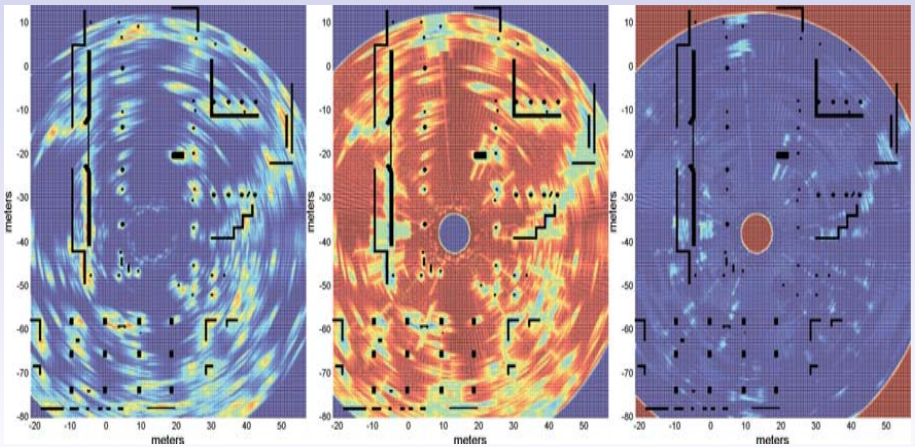


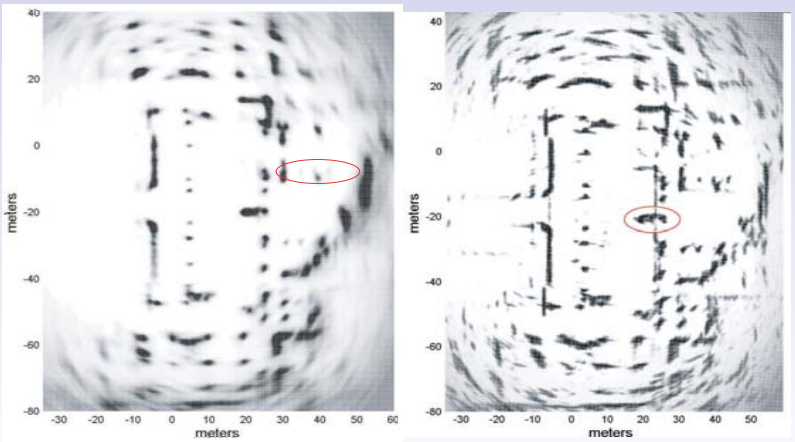
Figure: *Discrete probabilistic detection vs range likelihood NASSE comparison.*

Discrete Evidential Implementation



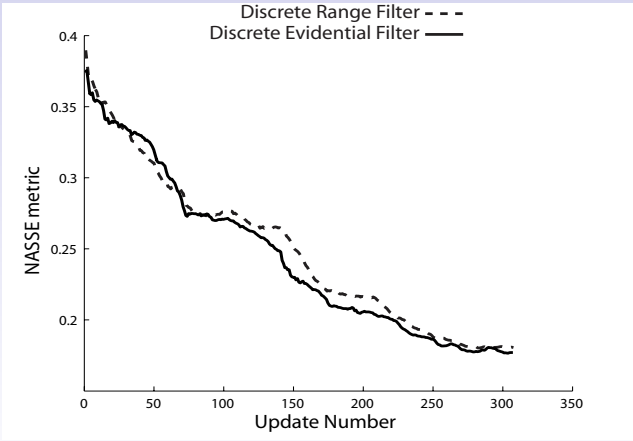
Instantaneous mass distributions on the map.

Discrete Evidential Implementation



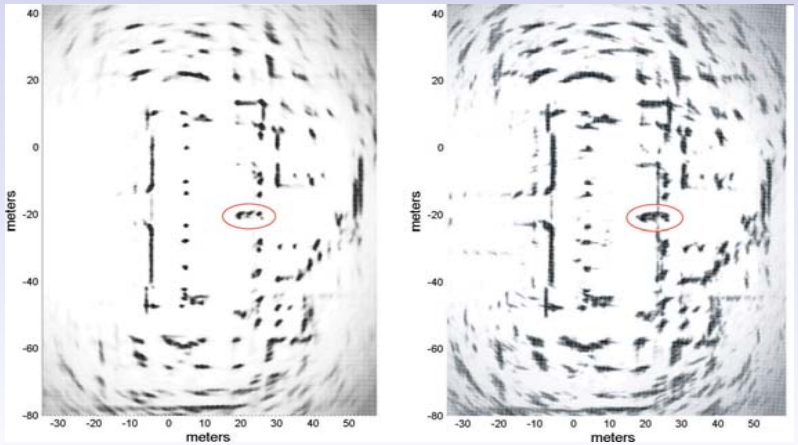
Discrete evidential detection filter (left) and discrete range filter (right).

Discrete Evidential Implementation



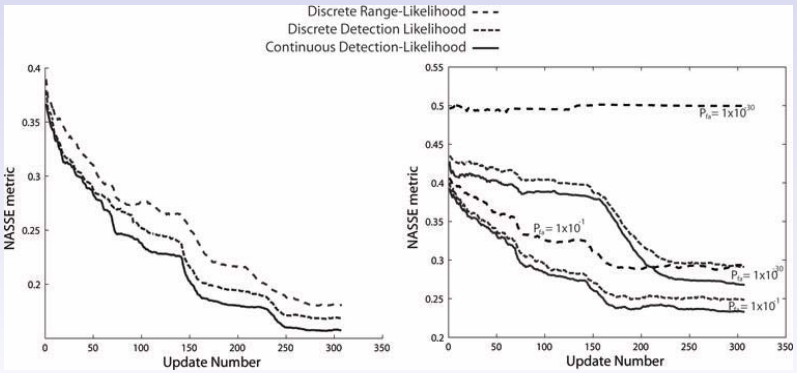
Discrete evidential filter NASSE.

Continuous Probabilistic Implementation



Continuous detection filter (left) and discrete range filter (right).

NASSE Comparisons



NASSE comparison.

NASSE Comparison vs. Detector Parameter

$$g_k(z_k = D | m_{k,(r)} = O, X_k) = \left(1 + \frac{T_r}{1 + \bar{\Re}_r}\right)^{-2W}$$

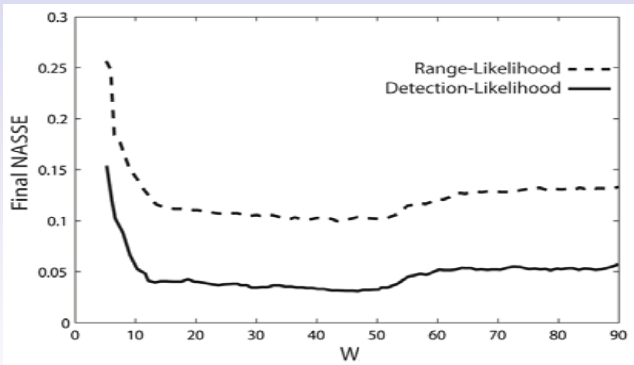


Figure: *Continuous detection vs range likelihood NASSE vs sliding window width.*

Campus Results

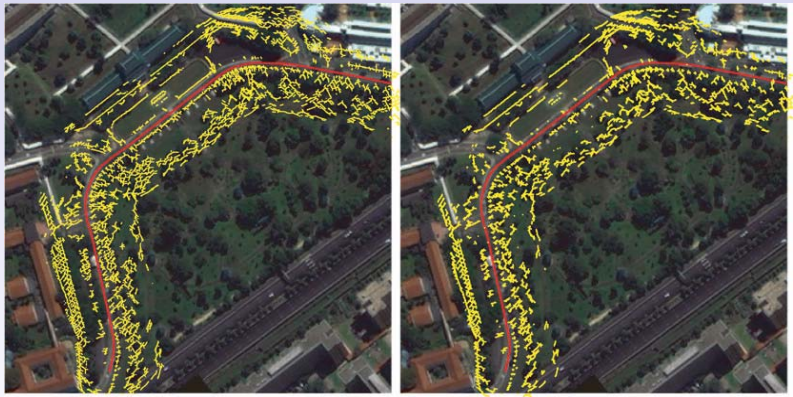


Figure: *Campus excerpt map estimate*

Campus Results

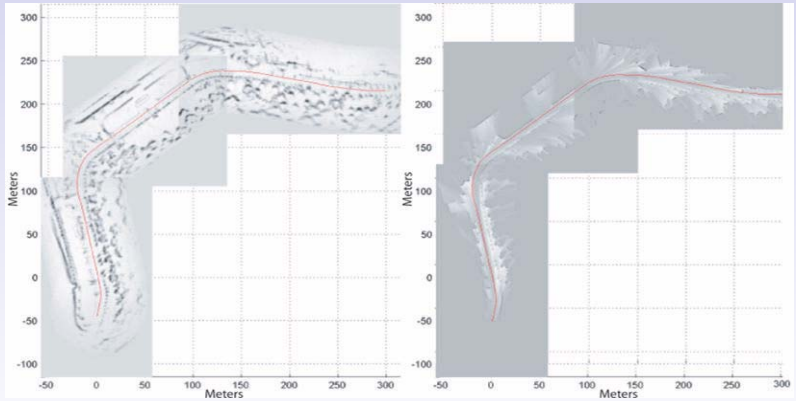


Figure: *Comparison of radar and laser posterior grid map estimates*

Campus Results

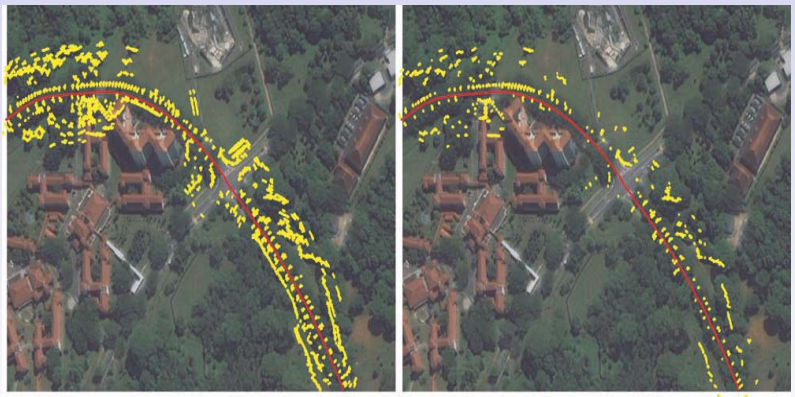


Figure: *Campus excerpt map estimate*

Campus Results

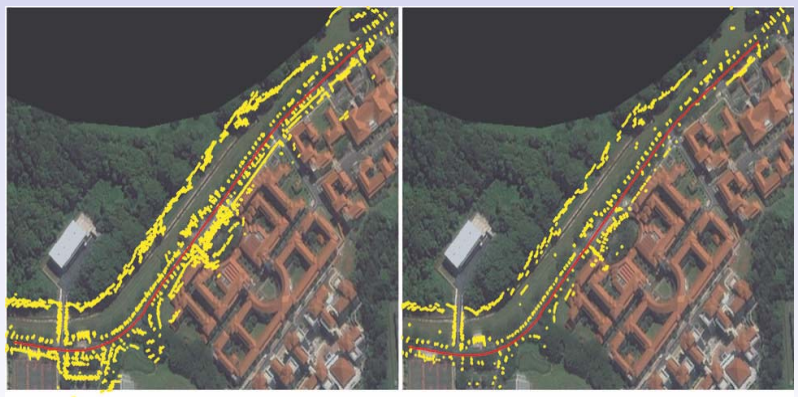


Figure: *Campus excerpt map estimate*

Presentation Outline

Conclusions

- Autonomous safety is highly dependant on accurate environmental representation
- ✔ Error of estimated grid maps can be reduced by incorporating the measurement uncertainty directly into the measurement likelihood
- ✔ Changing measurement space to detection/non-detection makes the likelihoods physically intuitive
- ✔ Likelihoods derived and mapping filters implemented using a MMWR sensor
- ✔ Improved mapping accuracy, particulary in situations of high false alarm and missed detection probability

Future directions

- Extend detection recursion to other sensing modalities
- For radar: Develop the EKF - Evidential Kalman Filter (Continuous evidential likelihoods)
- Extend to feature extraction algorithms - estimating the probability of feature existence

Acknowledgements

- The research described in this project was funded in part by the Singapore National Research Foundation (NRF) through the Singapore-MIT Alliance for Research and Technology (SMART) Center for Environmental Sensing and Monitoring (CENSAM).

Session III

SLAM, Localization, Reconstruction

- **Title: Experimental Comparison of Bayesian Outdoor Vehicle Localization Filters**
  Authors: Alexandre N. Ndjeng, Dominique Gruyer, Alain Lambert, Sébastien Glaser, Benjamin Mourllion

Experimental Comparison of Bayesian Outdoor Vehicle Localization Filters

Alexandre N. Ndjeng[1], Dominique Gruyer[1], Alain Lambert[2], Benjamin Mourllion[3], Sébastien Glaser[1]

[1]LIVIC
INRETS/LCPC
Bât 824, 14 route de la minière
78000 Versailles Satory - France

[2]IEF, UMR CNRS 8622
Université Paris Sud-XI
Bât. 220, Centre d'Orsay
91405 Orsay cedex - France

[3]MIPS-MIAM
Université de Haute Alsace
12 rue des Frères Lumières
68093 Mulhouse cedex-France

alexandre.ndjeng@inrets.fr

***Abstract*— Localizing a vehicle consists in estimating its state by merging data from proprioceptive sensors (inertial measurement unit, gyrometer, odometer, etc.) and exteroceptive sensors (GPS sensor). A well known solution in state estimation is provided by the Kalman filter. But, due to the presence of nonlinearities, the Kalman estimator is applicable only through some alternatives among which the Extended Kalman filter (EKF), the Unscented Kalman Filter (UKF) and the Divided Differences of 1^{st} and 2^{nd} order (DD1 and DD2). We have compared these filters using the same experimental data. The results obtained aim to rank these approaches by their performances in terms of accuracy, confidence and consistency.**

I. Introduction

In the intelligent vehicle applications, the extended Kalman filter (EKF) has unquestionably been up to now, the dominating state estimation technique [1], [2]. More recently, some new methods have been developed in order to improve the nonlinear estimation. They include some other variants of the Kalman filter such as the Divided Differences of first and second order (DD1 & DD2) [3], [4] and the unscented Kalman filtering (UKF) [5]. The principle of these new approaches is based on the linearization of the process and measurement functions by statistical linear regression functions, through sampling points in the region around the state estimatee.

Up to now there is not much work related to the comparison of performances of the more recent filters. A study carried out in [6] aims to evaluate some Kalman filter variants (EKF, DD1, DD2, UKF) capacities to linearize the process and measurement models. This work theoretically compares the filters' performance separately for the prediction step and the correction step, but does not analyze their overall performances. [7] shows the performances of these estimators (EKF, DD1, DD2 and UKF) in their predictive steps in road vehicles localization. The objective of this paper is to complement the work already carried out in [7] by taking into account the overall localization process (both the predictive and corrective steps). In order to ensure the comparison between these methods, two criteria of performance are set and applied. The first criterion concerns the evaluation of the accuracy of each method. We use a reference trajectory provided by a centimetric RTK GPS. The next studied criteria relate to the size of the 2σ scaled confidence envelops, the 2σ uncertainty ellipse areas and the Normalized Innovation Squared (NIS), from which each filter consistency will be deduced. This paper is organized as follows. Section 2 provides an overview of the Kalman Filter approaches (EKF, UKF, DD1, DD2) for nonlinear estimation. Section 3 presents the system modeling and the comparison criteria that we use in this work. In section 4, we describe the experimental environment and analyse the results of the 4 filters.

II. Review of the Kalman Filter Approaches for nonlinear Estimation

A. The Extended Kalman Filter (EKF)

The main difference between traditional and extended Kalman filter appears in the computation of the various matrices. In the KF, the process and the measurement matrices are composed of "true" linear functions; whereas in the EKF, these matrices (called Jacobian matrices) are composed of Taylor first order linearized functions.

Although the EKF has been shown reliable in many practical driving situations, it has some well known drawbacks. A major one concerns the hypothesis related to the point of the linearization. Theoretically, the nonlinear process function f is linearized around the true current state. But in the implementation, this function is linearized around the estimated value of X, leading to an additional error [7]. Moreover, another evident limitation of this filter concerns the possibility of computation of the Jacobian matrices. For very complex systems, strong nonlinearities can generate system instability problems; therefore the theoretical calculation of these matrices can simply become impossible, for example when the process or measurement functions are non-differentiable. However for intelligent vehicles applications, most of the used functions are differentiable, so the main drawback

concerns linearization. In order to bypass these limitations, some other methods based on a derivative free approach are presented in the following. These methods are shown often more powerful than the EKF and include UKF [5], DD1 & DD2 [4].

B. The Unscented Kalman Filter (UKF)

The UKF is an application of the unscented Transformation to a mean square recursive estimation [5]. This tranformation is a method for calculating the statistics of a random variable that undergoes a nonlinear process. We consider a random propagating variable X through a nonlinear function, $Y = f(X)$. A set of *sigma points* with mean $\bar{X}$ (mean of X) and covariance P_{xx} (covariance of X), is deterministically chosen. The nonlinear function f is applied to each point to yield a cloud of transformed points with statistics $\bar{Y}$ and P_{yy}. The n-dimensional random variable X is approximated by $2n+1$ weighted *sigma points*.
This transformation is built following the steps below:

- The sigma points are propagated through the nonlinear function. For $i = 0, \cdots, 2n$

$$\mathcal{Y}_i = f[\mathcal{X}_i] \tag{1}$$

- The mean is calculated as the weighted mean of the transformed points:

$$\bar{Y} = \sum_{i=0}^{2n} W_i \mathcal{Y}_i \tag{2}$$

  where W_i is the weight of the i^{th} point.
- The covariance is obtained according to Eq. 3

$$P_{yy} = \sum_{i=0}^{2n} W_i [\mathcal{Y}_i - \bar{Y}][\mathcal{Y}_i - \bar{Y}]^T \tag{3}$$

C. The Divided Differences Kalman Filter of first and second Order (DD1 & DD2)

These filters' formulations were proposed by Norgaard et al [4]. Both are based on the Stirling interpolation (presented in the following), and their implementation methods are very similar. In the DD1, we are limited to the 1^{st} order interpolation, and in the DD2 the functions are linearized at the 2^{nd} order. The DD1 & DD2 filters differ from the EKF in the fact that the Jacobian matrices are replaced by divided differences. Therefore the correction steps are the same. The main difference appears in the filters' covariance matrices update.
In the EKF we used to linearize the function from a Taylor series development of first order. In order to implement the Sterling interpolation two operators are defined. Let us consider a one dimensional interval with length ξ:

$$\delta f(X) = f(X + \frac{\xi}{2}) - f(X - \frac{\xi}{2}) \tag{4}$$

$$\mu f(X) = \frac{1}{2}(f(X + \frac{\xi}{2}) + f(X - \frac{\xi}{2})) \tag{5}$$

The Stirling interpolation formula is obtained by applying both the above operators to the mean value μ_X rather than directly to X. The second order interpolation gives

$$f(X) \simeq f(\mu_X) + f'_{DD}(\mu_X)(X - \mu_X) + \frac{f''_{DD}(\mu_X)}{2!}(X - \mu_X)^2 \tag{6}$$

where μ_X is the mean value of distribution X and

$$f'_{DD}(\mu_X) = \frac{f(\mu_X + \xi) - f(\mu_X - \xi)}{2\xi} \tag{7}$$

$$f''_{DD}(\mu_X) = \frac{f(\mu_X + \xi) + f(\mu_X - \xi) - 2f(\mu_X)}{\xi^2} \tag{8}$$

During the DD1 and DD2 filters implementation, a Householder triangulation is introduced, in order to compute the characteristic divided difference matrices. These matrices are then used in order to get the process and measurement noise covariance matrices, as well as the predicted and corrected state error covariance matrices. More details can be found in [4], [6], [7].

III. System modeling and Comparison criteria

A. Vehicle model and GPS model

All the filters that are compared are based on the kinematic model presented in equation 9.

$$\begin{cases} x_{k|k-1} &= x_{k-1|k-1} + \\ & V_k T cos(\psi_{k-1|k-1} + T\dot{\psi}_k/2) cos(\delta_k) \\ y_{k|k-1} &= y_{k-1|k-1} + \\ & V_k T sin(\psi_{k-1|k-1} + T\dot{\psi}_k/2) cos(\delta_k) \\ \psi_{k|k-1} &= \psi_{k-1|k-1} + \dot{\psi}_k T \end{cases} \tag{9}$$

where T is the time period, $[x_k, y_k]'$ is the position vector, V_k is the vehicle velocity and δ_k is the steering front wheel angle. This angle is used to take into account the kinematic constraints on the vehicle [8]. ψ_k and $\dot{\psi}_k$ represent the yaw angle (heading) and the yaw rate, respectively.
The GPS observation model is linear and is the same for all filters, with the corresponding observation matrix

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{10}$$

B. Comparison criteria

In ordre to evaluate and then compare the presented filter performances, many measures can be used. In this work, we focused on the accuracy measure, the confidence on the outputs and then the consistency of various filters.

- **Accuracy:** For our localization application it is more appropriate to use as accuracy measure, the Average Euclidean Error ***AEE*** [9] defined as

$$AEE(\hat{X}) = \frac{1}{M} \sum_{i=1}^{M} \sqrt{\tilde{X}_i^T \tilde{X}_i} \tag{11}$$

Where M is a number of trials, $\tilde{X} = X - \hat{X}$ is the estimation error, X is the real state vector and $\hat{X}$ is its estimatee. AEE is chosen because it better approximates the true mean error, that means the true average Euclidean distance between the real value and the estimatee.

- **Confidence in the output:**
  - Firstly, we focus on the 2σ confidence envelopes in order to evaluate the level of confidence we can have on each filter's output. Evaluating the various filters uncertainties can derive in the consistency of each filter's estimated error assessment.
  - Secondly, the 2σ uncertainty ellipse areas are considered. In fact, the results of the filters are given *a posteriori* with an uncertainty symbolized by an ellipse. In order to obtain the size of the axes of these ellipses, it is necessary to compute the eigenvalues of the covariance matrix P. These values are weighted with a factor $k = \sqrt{-2\log(1-P_a)}$, where P_a is the membership probability [10].
- **Consistency:** Various filters' consistency is studied. A state estimatee $\hat{X}$ with covariance matrix P is called consistent if it satisfies equation 12 [6], [11].

$$E\left[\left(X-\hat{X}\right)\left(X-\hat{X}\right)^T\right] \equiv E\left[\tilde{X}\tilde{X}^T\right] \leq P \quad (12)$$

The Normalized Innovation Squared (NIS) measure, ϵ, is used [11] to characterize the filters' consistency.

$$\epsilon = \tilde{z}^T S^{-1} \tilde{z} \quad (13)$$

$\tilde{z}$ is the filter innovation and S the innovation covariance matrix. Under the hypothesis that the filters are consistent and approximately linear-Gaussian, ϵ is χ^2 (chi-square) distributed with $dim(Z)$ degrees of freedom. The average value of the NIS ϵ then tends toward the dimension of the observation vector $Z = [x_{GPS}, y_{GPS}]$

$$E[\epsilon] = n, \quad with\ n = 2 \quad (14)$$

Therefore, a filter will be called consistent, if the average NIS is less or equal to 2.

IV. Experimental results

A. Test track and collected data

The tests carried out in this work used real data collected with It is assumed that the slope and bank angles remain negligible. The experimental data used in equation 9 were directly provided by an inertial measurement unit Crossbow VG400 (yaw rate $\dot{\psi}$), an odometer fixed to the front axis (vehicle speed V) and a coder that recorded the steering angle at the front wheel (δ). A low cost GPS directly provided correction data and initial states. The state noise was derived from various sensor noises. In our experiment, we have $\sigma_{odometer} = 0.005m/s$ and $\sigma_{gyro} = 0.05rad/s$. The complete track has a length of approximately $5.5km$. During the tests, the reference trajectory was obtained by using a fusion of a very accurate INS and a RTK[1] GPS (Thales).

During the first scenario (complete test track, see figure 1), all the sensors were synchronized thanks to GPS timestamps, and the system provided an output at the frequency of $5Hz$. In the area surrounded by trees (see figure 2 around $(400, -500)$) the GPS points have an error of more than $10m$ from the reference.

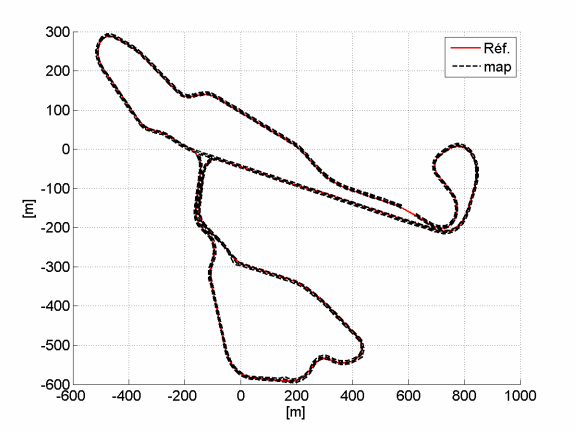


Fig. 1. Test track in scenario 1

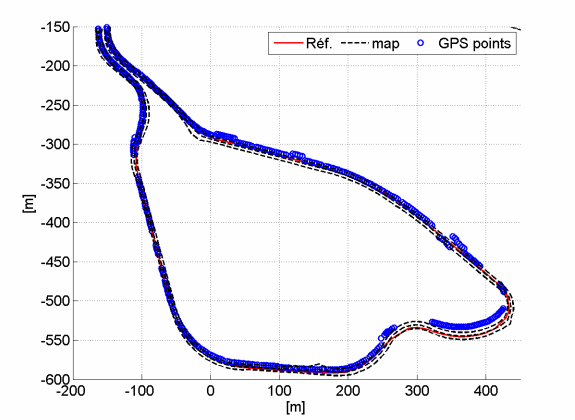


Fig. 2. Zoom on Forest Area in scenario 1

The second scenario took place on a part of the road track described in figure 3. During this scenario, the GPS sensor was switched off from time to time. During such periods, the localization was performed with only proprioceptive sensors and the filters ran in the predictive step. The update was performed only in presence of GPS.

B. Accuracy

1) Scenario 1: Figure 4 presents the positioning Euclidean errors for the 4 presented methods. The methods show rather comparable errors.Table I presents the mean

[1]Real Time Kinematic

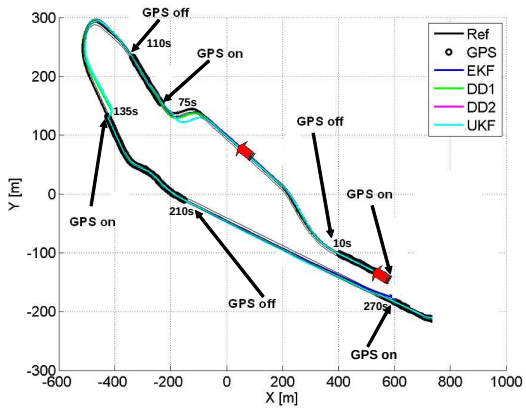


Fig. 3. Filters localization in scenario 2

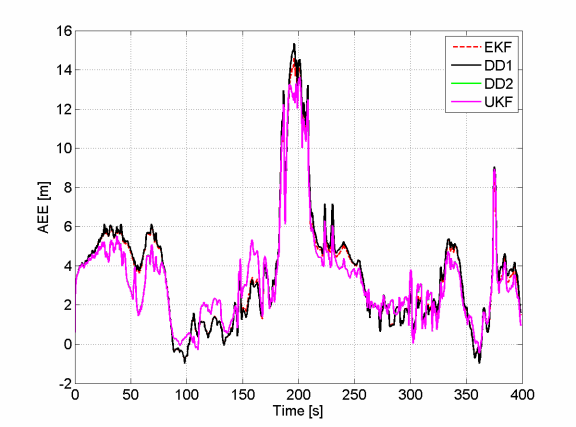


Fig. 4. Euclidean positioning error in scenario 1

and maximum values of the Euclidean errors. The analysis of this table brings a confirmation of what figure 4 indicates. The performances of the EKF, UKF, DD1 and DD2 filters are globally similar (2.91%difference). When we focus on these results, we can notice that EKF and DD1 have the same mean error $3.093m$, they appear globally more accurate than UKF and DD2 (mean AEE $3.186m$ and $3.185m$).
Nevertheless the max values of the error, which are reached in the area in figure 2 (from $125s$ to $220s$), reveal that the UKF and DD2 have a better behavior. This result contrasts with the global performance order established from mean AEE, the reason for this is in the following. In fact, this area is characterized by very poor quality GPS data combined with strong nonlinearities

TABLE I
MEAN AND MAX AEE IN SCENARIO 1

Method	Mean AEE (m)	Max. AEE (m)
EKF	3.093	15.274
UKF	3.186	14.732
DD1	3.093	15.273
DD2	3.185	14.732

in the prediction (strong turns in the trajectory). In such situations filter outputs are more influenced by the predictive step. UKF and DD2 better handle such nonlinearities and are less influenced by updating GPS data. Considering Table I, the second order filters (UKF and DD2) are slightly more accurate (respective max AEE $14.732m$) than first order filters (EKF max AEE $15.274m$, DD1 max AEE $15.273m$) in such a situation. But in each group the filters behave very similarly.
To conclude, it appears that following given situations, EKF and DD1 have better performances than UKF and DD2 and vice versa. But globally, first order filters are recommended for such a scenario.

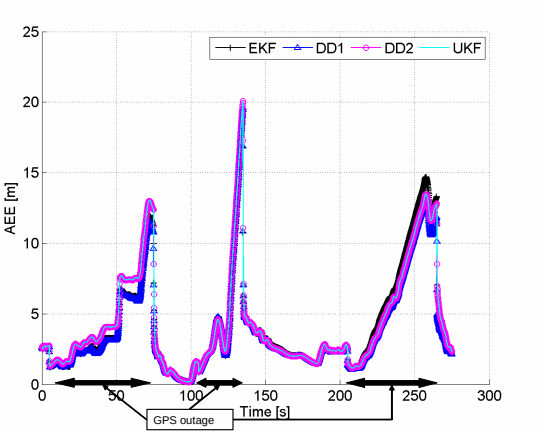


Fig. 5. Euclidean positioning Error in scenario 2

2) scenario 2: The Euclidean positioning error in this scenario is shown in figure 5. First of all, it is remarquable that in presence of GPS signal, all the methods show rather comparable errors (like in scenario 1). Important differences appear when GPS is switched off. This is an interesting illustration of the theory : in fact, these filters are theoretically different in the manner they handle the nonlinear process fonction. These theoretical differences are presented in [6] and [7].
In this scenario, the common GPS observation model is linear. Thus, the correction stage will not show important differences between filters. However, during the GPS outages, figure 5 confirms that EKF and DD1 (first order filters) react similarly and are more accurate than UKF and DD2, which are also similar.
During the first GPS outage, EKF and DD1 mean AEE ($4.89m$ and 4.73) is almost $1m$ lower than DD2 and UKF mean AEE($5.73m$ and $5.75m$). The maximum AEE has the same order. During the second GPS outage, the AEE differences are reduced to about $15cm$ for the mean values and $50cm$ for the max. But the first order filters are still more accurate. Figure 6 presents the X-axis and Y-axis positioning error. It reveals that the difference on the X-axis is more important than on the Y-axis. The explanation is found in the modeling error and the cumulated error of second order filters DD2 and UKF. In presence of trajectory non linearity, these filters react as

TABLE II

MEAN AND MAX AEE DURING GPS OUTAGES (SCENARIO 2)

Method	GPS outage 1 $[10-75s]$		GPS outage 2 $[110-135s]$		GPS outage 3 $[210-270s]$	
	$meanAEE(m)$	$maxAEE(m)$	$meanAEE(m)$	$maxAEE(m)$	$meanAEE(m)$	$maxAEE(m)$
EKF	4.89	11.95	4.87	19.53	7.58	14.50
$DD1$	4.73	11.56	4.88	19.59	6.68	13.23
$DD2$	5.73	13.05	5.02	20.08	6.96	13.48
UKF	5.75	13.00	5.02	20.08	6.93	13.48

if the vehicle turns earlier than it should be (see figure 3). In fact, the bias between the first order filters and the second order filters comes from the way the estimatee is computed. Using velocity and heading, the primary computation is done in the polar space and the result is returned in the Cartesian space. This polar to Cartesian conversion problem was handled in [7], and it derives in a cumulative error for second order filters, which is visible on figure 3.

During the third GPS outage, the filters still react similarly with very close errors, except for EKF which is $1m$ worse. Examining figure 6, we can see that all the filters deviate progressively on the Y-axis (from $210s$). However, EKF deviates more than the others. Considering that the road configuration is linear here, this deviation comes from the sensors noise modeling (gyro) and the inadequate initialization before the GPS was switched off. The results given in table II reveal that, considering the defects cited above (sensors noise and inadequate initialization), DD1 is more accurate and shows a better robustness than EKF and second order filters in such situations.

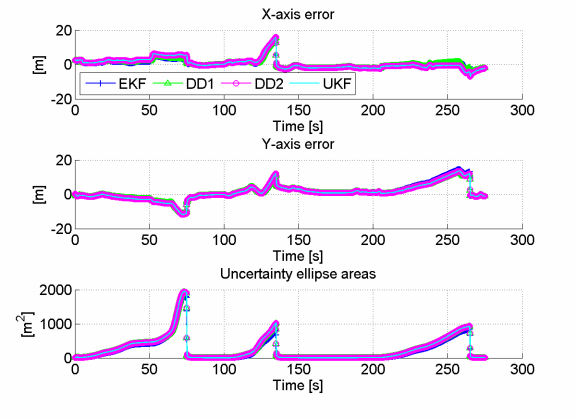


Fig. 6. X-axis (top) Y-axis (middle) positioning error and 2σ uncertainty ellipse areas (bottom)

C. Consistency

1) Scenario 1: In order to study the confidence in those estimatee, we compute the 2σ scaled envelopes provided by each filter (95% probability region). On figures 7 and 8 we superposed the positioning axis error

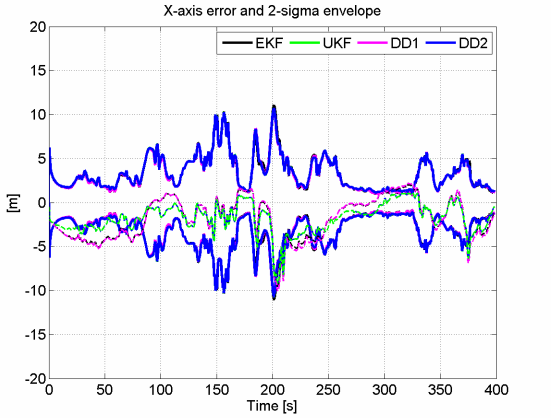


Fig. 7. X-axis 2σ envelope and error in scenario 1

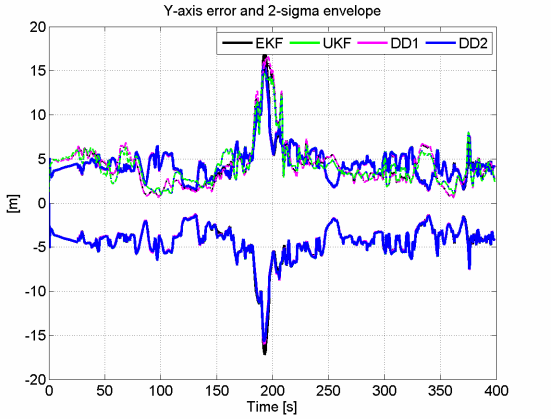


Fig. 8. Y-axis 2σ envelope and error in scenario 1

and the associated 2σ envelope. For all the filters, we can see that the estimation error on each axis is most of the time inside the corresponding envelope. This means that globally, the error estimatee given by the filters through their covariance matrices is consistent with the real positioning estimation error. The forest area is an exception. With poor quality GPS data, filter envelopes grow considerably (see figures 7 and 8), deriving in a loss of confidence in the estimation. However all the filterenvelops are almost identical.

Figure 9 shows the NIS of the position innovation normalized to a 95% probability region, assuming a χ^2 distribution. Most of the time the normalized NIS of EKF, UKF, DD1 and DD2 are below 2.0. This means that

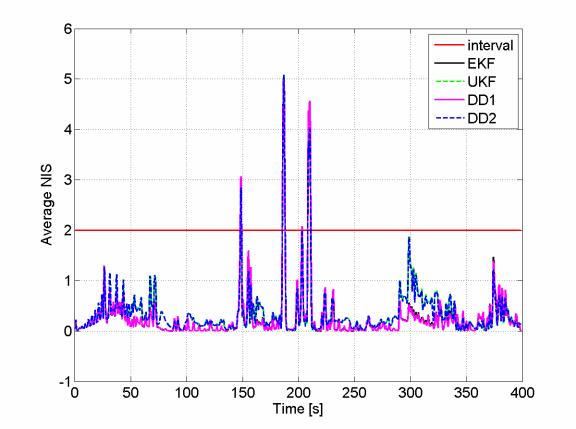


Fig. 9. Filters NIS in scenario 1

the estimated filter uncertainties are most of the time consistent with the true estimation error, considering 95% probability region. Once more an exception appears in the forest area (from $125s$ to $220s$), where NIS values are often above 2.0. This last measure confirms that of the 2σ envelopes: the four filters behave similarly, the differences between their errors and their uncertainty are very small.

2) Scenario 2: The confidence of the presented filters output is tackled under the 2σ uncertainty ellipses analysis. Figure 6 (bottom) shows the filters ellipse areas. It reveals that the uncertainty ellipse areas grow considerably during GPS outages. During the first outage period $[10-75s]$, these areas almost reach to $2000m^2$. During the other GPS outage periods the maximal areas are around $1000m^2$. The behavior of various filters is very similar: the uncertainty is low when correction data are used, and high if not.

The uncertainty growth is more important during the first GPS outage. This part of the scenario is characterized by multiple trajectory nonlinearities combined with a long period without correction. During the second GPS uncertainty, the trajectory is highly non linear, and the outage lasted $25s$. The growth here is comparable to that during the third outage (linear, $60s$). These observations reveal that the uncertainty ellipse areas growth is also a fonction of the system non linearity: the stronger the nonlinearity, the stronger the growth.

V. Conclusion

In this paper an experimental comparative study of 4 Kalman based localization approaches (EKF, UKF, DD1, and DD2 filters) were presented. Previous works [6], [7] exhibit major differences as well as similarities between those filters. [6] shows theoretical differences whereas [7] analyses practical differences during the prediction steps. Nevertheless practical experiments taking into account the whole localization process (both prediction and correction steps) exhibit minor differences. The differences observed during the prediction step in terms of accuracy and uncertainty (due to the linearization or the use of the sigma points) are strongly reduced during the correction step.

According to our experiments, the choice of a given filter will depend on the situation: this means, the presence and the quality of correction data, or the presence of strong nonlinearities on the trajectory. Therefore

- if there is no GPS signal outage for a long time during the navigation, or if the GPS signal is of good quality (if the correction step runs efficiently), then it is not easy to propose a favorite filter. The use of one or another among the presented filters brings almost insignificant amelioration in terms of accuracy, as well as the confidence of each estimatee.
- if the GPS signal is of poor quality, in presence of strong nonlinearities we should consider UKF or DD2 which seem more robust in such situations.
- if the GPS signal is absent for a long time, the EKF or the DD1 are recommended as mentioned in [7], but our study also reveals that DD1 remains, to our best tunings, the most robust filter among the four.

In further work, these filters will be compared to other estimators which are theoretically assumed more robust, such as multiple model filters or Monte Carlo filters.

References

[1] I. Abuhadrous, F. Nashashibi, and C. Laurgeau, "3-d land vehicle localization: a real-time multi-sensor data fusion approach using rtmaps," in *11th International Conference on Advanced Robotics*, (University of Coimbra, Portugal), June 30th-July 3rd 2003.

[2] J. L. et *al.*, "Multisensorial data fusion for global vehicle and obstacles absolute positioning," in *IEEE Intelligent Vehicles Symposium*, (Columbus, OH, USA), June 9-11 2003.

[3] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *In IEEE Transaction on Automatic Control*, vol. 45, May 2000.

[4] M. Norgaard, N. K. Poulsen, and O. Ravn, "Advances in derivative-free state estimation for nonlinear systems," revised edition of the technical report imm-rep-1998-15, Technical University of Denmark, 2000.

[5] S. J. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *SPIE Aerosense International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, (Orlando, Florida), September 1997.

[6] T. Lefebre, H. Bruyninckx, and J. D. Schutter, "Kalman filters for nonlinear systems: A comparison of performances.," *The International Journal of Control*, vol. 77, May 2004.

[7] B. Mourllion, D. Gruyer, A. Lambert, and S. Glaser, "Kalman filters predictive steps comparison for vehicle localization," in *International Conference on Intelligent Robots and Systems*, IEEE/RSJ, 2005.

[8] A. Kelly, "Some useful results for closed-form propagation of error in vehicle odometry," cmu-ri-tr-00-20, Robotics Institute, Carnegie Mellon University, 2000.

[9] X. R. Li and Z. Zhao, "Practical measures for performance evaluation of estimators and filters," in *Workshop on Estimation, Tracking, and Fusion - A Tribute to Yaakov Bar-Shalom*, (Monterey, CA), May 2001.

[10] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. Journal of Robotic Research*, vol. 5, no. 4, winter 1986.

[11] Y. Bar-Shalom and X. R. Li, *Estimation and Tracking: Principles, Techniques, and Software*. Boston, MA: Artech House, 1993. Reprinted by YBS Publishing, 1998.

Session III

SLAM, Localization, Reconstruction

- **Title: Predictive Lane Detection for Simultaneous Road Geometry Estimation and Vehicle Localization**
  Authors: Chenhao Wang, Zhencheng Hu, Tomoki Maeda, Naoko Hamada, and Keiichi Uchimura

Predictive Lane Detection for Simultaneous Road Geometry Estimation and Vehicle Localization

Chenhao Wang, Zhencheng Hu, Tomoki Maeda, Naoko Hamada, and Keiichi Uchimura

***Abstract* — this paper describes a predictive lane detection method with assistance of road geometry data from digital road map to simultaneously estimate road shape and vehicle localization. In our approach, visual information is not the only source to detect lane and estimate road parameters, the road geometry information derived from digital road map has also been providing important predictive cues for lane detection. Comparing with the conventional vision-only based approaches, our system is able to provide more reliable and stable road geometry estimation result. In addition, a precise longitudinal localization can also be achieved through the piecewise polynomial matching algorithm. Simulative and real road tests under various environmental conditions have shown the effectiveness of the proposed method.**

I. Introduction

WITHIN the last two decades, lane detection is one of the primary research topics in advanced driving assistance systems (ADAS). Lane detection primarily works for vehicle's lateral control systems, namely, Lane Departure Warning System (LDWS) and Lane Keeping Assist System (LKA) to estimate vehicle position and posture relative to road lane. Although some works utilize LIDAR or RADAR to detect roadside boundary, or uses embedded magnetic markers in the road way, the predominant approach by far is the use of video camera and image processing to extract the land and road edge markings from the image – exactly what human drivers do in visually processing the road scene.

So far, exiting works in vision based lane detection literature generally refer to a lane model consisting of camera model and road shape model. Most of the approaches are conduced to calculate related road shape parameters and trace lane boundaries in a recursive prediction-updating loop. Widely adopted methods for road parameters estimation are Kalman Filtering (KF), Extended Kalman Filtering (EKF) and Particle Filtering (PF) techniques. Road model is another argumentative topic in the field of lane detection. Straight line is the simplest road shape model, but is also erroneous for non-straight or non-flat road. Recently most approaches prefer to use piecewise line segments with the assumption of flat road surface, such as piecewise lines, circular [3], clothoid [4], polynomial approximation and so on. Although some more precise models like three-dimensional (3-D) road shape [5] could gain accuracy describing lane variance in both horizontal and vertical, it surfers from the high computational cost and high sensitivity to noise on the contrary.

The most difficult challenge for vision-based approaches is the robustness to different environment condition. Various meteorological and lighting conditions (day, night, sunny, rainy, snowy), road environmental conditions (occlusion, degraded road markings) significantly influence the estimation results. Most of the previous works depend on occupancy rate of proposed road model points and actual road features (edges or lane markers) on the image to evaluate result's accuracy. However, even with high occupancy rate, the estimated road shape model may not be accurate if two or more parameters are problematic. For example, on a curve road, if road width is estimated wider than its true value and road curvature is slightly bigger, we can still observe the image matches perfectly with the estimated road shape model.

To solve the problems mentioned above, a Predictive Lane Detection (PLD) algorithm is proposed in this paper. PLD is a hybrid solution composing of prediction module and visual detection module. The prediction module for road geometry is estimated by vehicle localization and road network, because it performs more reliable and robust for road prediction than the previous vision-only based approaches (example shown in Fig.1). In addition, prediction is also used for lane tracking on noisy image by limiting the detection zone on the image close to camera.

Fig. 1: The result of road geometry estimation projected on real scene without visual detection in stormy weather

C.H. Wang, Z. Hu, and K.Uchimura and are with the Graduate School of Science and Technology, Kumamoto University

Graduate School of Science and Technology, Kumamoto University, 2-39-1, Kurokami, Kumamoto, Japan, 860-8555

(e-mail: wang@navi.cs.kumamoto-u.ac.jp, hu@cs.kumamoto-u.ac.jp uchimura@cs.kumamoto-u.ac.jp and shun@navi.cs.kumamoto-u.ac.jp,).

The paper is outlined as follows: section II describes our approach for PLD with constrains, road model and algorithm procedure. Section III discusses road geometry estimation in detail relying on present digital map. And Section IV introduces visual detection modules and how to analyze the parameters in each module. Hybrid results are presented in the finally section.

II. OUR APPROACH

A. Constrains

In our project, we utilize 2-D digital road map to reconstruct front road geometry for lane detection. And roadside variance in vertical plane is ignored here. In order to achieve our goal for PLD, we develop an image-based approach by taking following constrains into account.

i. It must collaborate with vehicle localization module, which is required for high accuracy and real-time performance.
ii. Local road network (2-D map, 3-D map) should be included unless a more precise approach could support for road information in detail.
iii. In our approach, Vehicle Coordinate System (VCS) and Camera Coordinate System (CCS) are regarded as same coordinate system, because we set GPS's antenna, Gyroscope and camera in the same plane.
iv. The algorithm in this paper assumes in a horizontal plane and the road's vertical curve is ignored. But the vertical part could also be recovered as long as 3-D map employing in system.
v. The rolling angle of vehicle is set as constant value.

B. Road Model

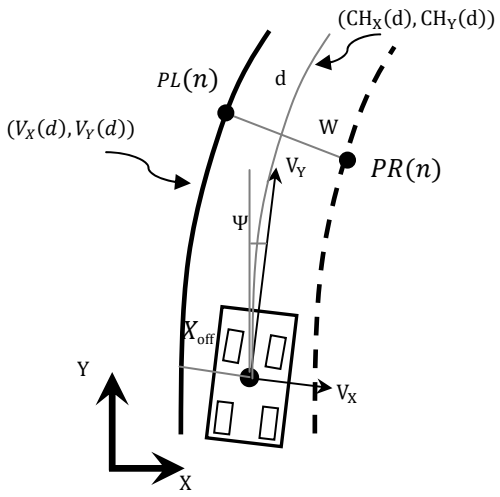


Fig. 2: Relationship between vehicle and road from bird's-eye view

Since road model could figure lane mark's position and its variety in VCS (see Fig. 2), how to define a road model precisely is a key problem for most lane detection. In our approach, we could replace lane's variety part with road geometry estimation. As the original reference information (road width and vehicle's offset) is also included in road model, road model for PLD is defined as following equation:

$$V_X(d) = \pm 0.5W + X_{off} + CH_X(d) \qquad (1)$$

- $V_X(d)$: is the position in latitudinal direction of VCS;
- W: is the width of lane;
- X_{off}: is the lateral displacement in vehicle coordinate system;
- $CH_X(d)$: is lane variety in latitudinal direction of VCS;
- d: is the distance along the road network;

$CH_X(d)$ is a crucial part driven from road geometry. Comparing to other models by visual detection, our approach utilizes the reference points from road network in 2-D plane directly, which should be more reliable than points of image-based. The road geometry is recovered relying on vector of distance. It could simulate road geometry not only in latitudinal direction but also in longitudinal direction which used to be ignored by many approaches. Besides, lane width and lateral displacement are detected by visual detection module.

C. Algorithm Procedure

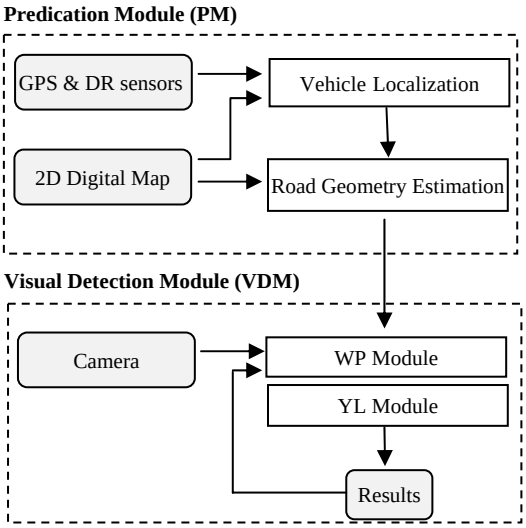


Fig. 3: Procedure chats of Predictive Lane Detection

Figure 3 presents the system block diagram. In prediction module, vehicle localization is estimated based on vehicle motion model with supporting with GPS and DR sensors. Digital road map is not only applied Map Matching (MM), but also provides node points of road network. In the step of road geometry estimation, new reconstruction approach based on route distance is designed for various road networks. Before visual detection, the PM could offer information of road geometry as soon as possible.

In Visual Detection Module (VDM), system will refer to

the information from PM first. It will help to confirm the processing area avoiding noise's influence. And the error causing by road geometry should be considered in PW (vehicle Pitch angle and road Width) module and YL (vehicle Yaw angle and vehicle Lateral displacement) module. These two modules are proposed to analyze feature points for related parameters.

III. PREDICTION MODULE

In section II, we have mentioned the road model (Equ.1) as crucial part for lane detection. Lane variety used to be recovered by image-based points with various models, such as circular, clothoid or polynomial ways. Because road parameters are very sensitive from reference points through image processing, we try to rebuild road geometry by utilizing vehicle localization and digital map. It is supposed to be a robust way to acquire the road parameters reliable and stable.

Since vehicle localization and digital map were introduced by many papers [7], [8], [9], it is no necessary to discuss these technologies in detail but know about the information from this module. Position and orientation $(v_x(k), v_y(k), v_ori(k))$are given by localization system with time sequence **k.** Road node points (r_x, r_y) consisting for road network are provided by digital map.

A. Road Geometry Reconstruction

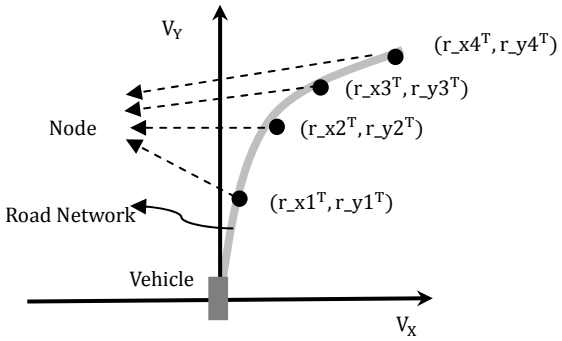


Fig. 4: Road geometry reconstruction in VCS

First of all, the node points should be transformed into VCS (see fig. 4). Vehicle always locate at origin point by tangent with road network. Node points in front of car, could figure out a general road's direction within certain distance. Polynomial method is chosen to approximate to real road because it is flexible to reconstruct any kinds of road. If road points by a selected route, we could simulation any road geometry by polynomial method. Furthermore, road geometry will be changed with vehicle following on the road network. So this method could express the road geometry precisely and timely.

The traditional method for polynomial is set up a relationship between V_x and V_Y directly. It relies on the order of polynomial and the number of node points. So it is hard to recover geometry accurately especially in some situations such as intersection and U-turn.

Here, we consider the polynomial in low-order and route distance is chosen as variety vector for recover V_x and V_Y independently. This method could simulate geometry as to practical situation and be realized easily by the way of mean-square. Although computational expense for $CH_X(d)$ and $CH_Y(d)$ estimation are increases for double, it could recover the geometry in a low-order and high precision.

$$CH_X(d) = \sum_1^n A(i)d^i \quad (2)$$
$$CH_Y(d) = \sum_1^n B(i)d^i \quad (3)$$

- $A(i)$: is the polynomial parameters in latitudinal direction of VCS;
- $B(i)$: is the polynomial parameters in longitudinal direction of VCS;
- d: is the distance along the road network;

The road network near to intersection is selected here and figure 5 expresses the procedure of our approach for road geometry estimation. Fig. 5(a) is the original data of vehicle localization (△) and road node points (*) in local coordinate system. And fig.5 (b) shows the result in VCS after coordinate transformation. According to node points (*) in fig.5 (b) projected into two feature spaces independently with the variety of distance, fig. 5(c) and fig. 5(d) is the feature space of x-d and y-d. The reconstruct route is shown as pink curve in the figure through proposed equation 2, 3 with 3 orders estimation. And the final combination result for road geometry is shown in the fig. 5(e) as green curve.

B. Prediction for Visual Detection

The purpose of this section is to provide the precise information for visual detection. In the section II, lane variety in road model is supposed to be given by road geometry estimation. And it could refer to the result of $CH_X(d)$ by equation 2. Furthermore, it could also be utilized in visual detection to eliminate influence causing by $CH_X(d)$ in latitudinal direction.

Besides, information $CH_Y(d)$ by equation 3 is also very important. However, it is ignored in the most situations because $CH_Y(d)$is considered as same as distance, which means visual detection model would like to select the feature points by horizontal line. According to point PL(n) and point PR(n) shown in figure 2, these two points are not at same horizontal line in VCS. That is why the road width of curve part is wider than straight one.

The slope k(d) of road geometry is expressed by equation 4. We could estimate the difference between left line and right line in longitudinal direction while the road width is considered the same, like the point PL(n) and point PR(n) in the figure 2. If we refer the right line as the baseline $CH_Y(d)$, the corresponding pair points should locate at

$CH_Y(d) + \Delta y(d)$ of longitudinal direction. So $\Delta y(d)$ could be expressed as equation 5.

$$k(d) = \frac{\partial CH_Y(d)}{\partial CH_X(d)} = \frac{\sum_1^n i*B(i)d^{i-1}}{\sum_1^n i*A(i)d^{i-1}} \quad (4)$$

$$\Delta y(d) = W \cdot \sin(\tan^{-1}(\frac{-1}{k(d)})) \quad (5)$$

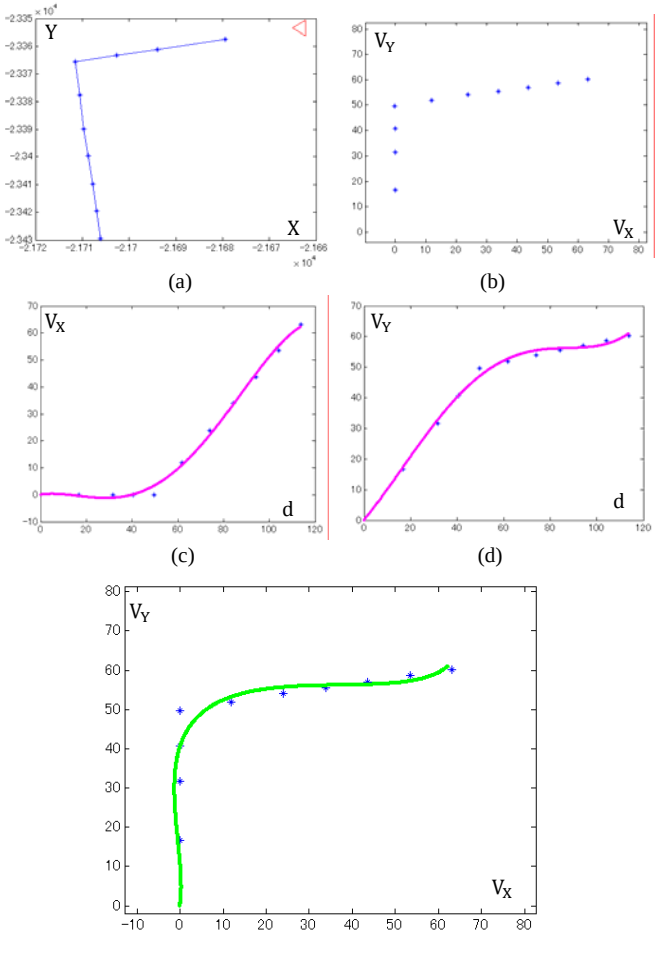


Fig. 5: Simulation results of road geometry reconstruction

IV. Visual Detection Module

Here we set the right line as the base line and right road line is denoted as equation 8, 9. Left road line could be denoted as equation 6, 7 according to road geometry estimation provided by PM.

$$LX(d) = f_x\left(\frac{-0.5W + X_{off} + CH_X(d)}{CH_Y(d) + \Delta y(d)} + \Psi\right) \quad (6)$$

$$LY(d) = f_y\left(\frac{H}{CH_Y(d) + \Delta y(d)} + \theta\right) \quad (7)$$

$$RX(d) = f_x\left(\frac{0.5W + X_{off} + CH_X(d)}{CH_Y(d)} + \Psi\right) \quad (8)$$

$$RY(d) = f_y\left(\frac{H}{CH_Y(d)} + \theta\right) \quad (9)$$

- $(LX(d), LY(d))$: is the point on the image of left lane marks;
- $(RX(d), RY(d))$: is the point on the image of left lane marks;
- $(CH_X(d), CH_Y(d))$: is the road geometry in VCS;
- H: is the height of camera;
- Ψ: is the yaw angle of car;
- θ: is the yaw angle of car;
- (f_x, f_y): is the focal length of the camera;

A. PW Module

PW module shows the linear relationship between vehicle Pitch angle and road width. The difference function (Equ. 10) between right line and left line is acquired by equation 6, 8, where $\Delta y(d)$ is set as zero.

$$\Delta X(d) = RX(d) - LX(d) = \frac{f_x W}{f_y H} LY(d) - \frac{f_x W}{H}\theta \quad (10)$$

$\Delta X(d)$ means the difference between pair points locating on right line and left line and $LY(d)$ is the related point on the vertical direction of image. Normally, $\Delta y(d)$ is close to zero and $\Delta X(d)$ could be estimated by same horizontal line. According to equation 10, the relationship of $\Delta X(d)$ and $LY(d)$ is the linear if two lines are parallel in vehicle coordinate system unless some places such as fork or junction. PW module is relying on this linear relationship and related parameters could be estimated by feature points on the space of $\Delta X(d)$ and $LY(d)$. But it should refer to $\Delta y(d)$ for picking up pair points on a significant curve.

Figure 6 is a typical image after feature extraction. Firstly, we should calculate the difference of feature points based one same horizontal line, which appear in zone belonging to left line or right line. And according to equation 10, the difference is calculated by given space of W and θ. So if we set up a series of reasonable W and θ, the different results could estimate the matching points with actual points from image. Figure 7 shows the matching probability in the space of W and θ. The peak area (shown as red points in fig 7) represents a reliable area with high probability. Finally width and pitch are estimated by statistical results.

The result is given in figure 8, blue points are the actual different points and pink points are given by estimated parameters. Of course, the difference points in figure 8 perform a linear characteristic of PW module. In this way, the lane width could be calculated by the slope of straight line and pitch angle is calculated afterwards by intercept.

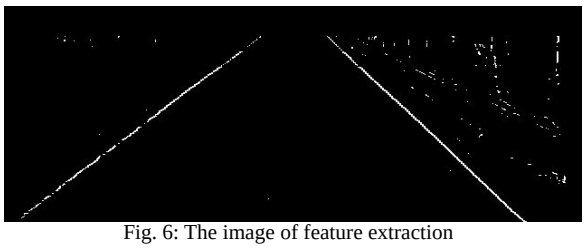


Fig. 6: The image of feature extraction

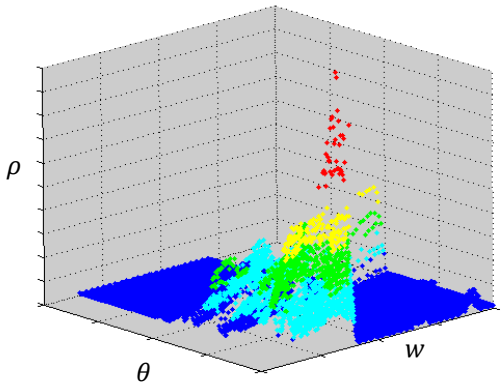


Fig. 7: Matching probability corresponding to W and θ

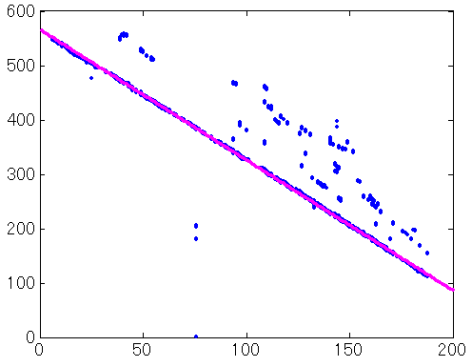


Fig. 8: Feature distribution map of PW module

B. YL Module

YL module is designed to get vehicle lateral displacement and yaw angle by referring to equation 6 or 8. The same as PW module, the parameters in this module should be seen as linear characteristic according to related functions. But we have discussed lane variety $CH_X(d)$ of latitudinal direction in section 3, which is the prediction vector to eliminate the influence caused by curve part. If the feature points on image are compensated in opposite direction, YL module could search maximal matching probability on image in linear way as PW module. For example, road geometry is estimated by prediction model (see fig. 9). Then it is transformed into image space through camera's parameters. So the original feature points (white points) are compensated by relevant compensation, which are shown as the green points in figure 10.

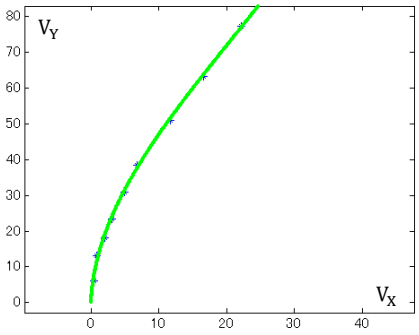


Fig. 9: Road geometry Estimation in VCS

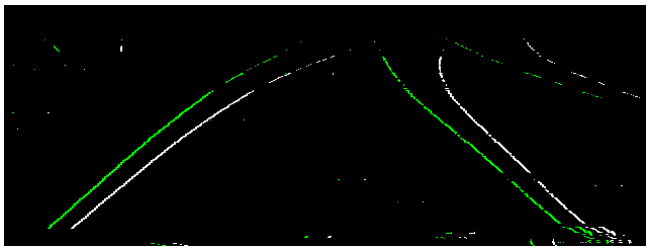
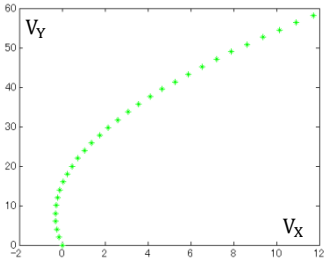


Fig. 10: Feature compensation based on geometry estimation for linear transformation

V. SIMULATIONS AND RESULTS

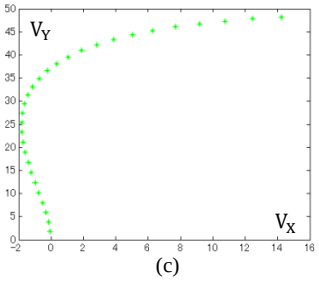
Our tests were based on the on-line data collection based on several sensors: a Teli CCD COLOR CAMERA was mounted on the front roof of test vehicle, image sequences were captured in NTSC format at the frame rate of 30fps, GPS data (Pioneer® GPS-M1ZZ) and inertial data (Gyroscope: Datatec®GU-3024 & Nissan LAFESTA CAN Speed) were sent to PC's serial port and recorded at the frequency of 1Hz and 60Hz separately; Shobunsh® Super Mapple ver.6 (1/25,000) was used as the 2D road map. Unfortunately, so far the algorithm has not been implemented by on-line processing. We just record data of vehicle localization and road network with synchronization of video recorder. And all tests are realized in the laboratory.

We have discussed the prediction module in section III and the result of road geometry estimation is shown in figure 5. Firstly, road geometry in VCS should be confirmed if it matches to the road on image or not. The range of geometry results ('*') in fig. 11(a) is 60 meters with interval of 2 meters. According to fig. 11(a), $CH_X(60)$ is 11.73 meters and $CH_Y(60)$ is 58.2. Although the error by longitudinal direction in VCS could be ignored, $CH_X(d)$ by latitudinal direction must be counted for visual detection. But in fig. 11(c), $CH_Y(d)$ could not be ignored as same as $CH_X(d)$. And the projected road by estimation results is shown in fig. 11(b) (d).



(a)

(b)



(c)

(d)

Fig. 11: Road geometry estimation and projection on image

As a result of geometry estimation shown in fig. 12 (a) (c) (e), other parameters are detected by visual detection module introduced in section 4. And all the parameters are included in table 1. Here we choose the 3-order of polynomial for reconstruction road geometry. Furthermore, final lines are plotted on the image based on parameters (shown in fig. 12 (b) (d) (e)). Although the results on the image match to real line mark, there are little excursion between proposed line and real line. The error by vehicle localization, digital map or road reconstruction model might cause the excursion.

VI. CONCLUSION

In this paper we proposed a predictive lane detection method with road geometry estimation that relying on precise localization and digital map. Although some constrains are defined for practical application, this method effectively estimate the reliable parameters and works well in different kinds of condition.

The online processing of this approach is still under evaluation and we are focusing on improving the accuracy and stability of vehicle localization. In addition we are also working on some important applications using this approach, like the classification of different types of lane marks based on different lane width distribution.

REFERENCES

[1] Z. Hu, K. Uchimura, "Fusion of Vision, GPS and 3D Gyro Data in Solving Camera Registration Problem for Direct Visual Navigation", International Journal of ITS, vol. 4, No. 1, December 2006

[2] [2] J. W. Lee, "A machine vision system for lane-departure detection," Comput. Vis. Image Undrest., vol. 86, no.1, pp. 52-78, Apr. 2002.

[3] K. Kluge and S. lakshmanan, "A deformable template approach to lane detection," in Proc. IEEE IV., Detroit, MI, 1995.

[4] E. D. Dickmanns and B. D. Mysliwetz, "Recursive 3D road and relative ego-state recognition," IEEE trans. Pattern Anal. Mach. Intell., vol. 14, no. 2, pp. 193-213, Feb. 1992.

[5] K. Kanatani and K. Watanabe, "Reconstruction of 3-D road geometry from images for autonomous land vehicles," IEEE Trans. Robot. Automat., vol. 6, pp. 127-132, Feb.1990.

[6] Xiong D., "A Three-Stage Computational Approach to Network Matching", Transportation Research C, vol. 8, pp. 13-36,2000.

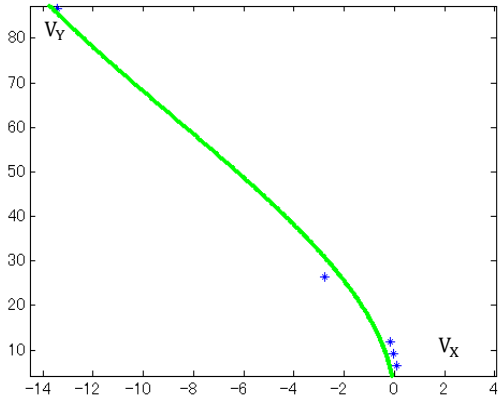
[7] Bernstein D., and Kornhauser A., "An introduction to map matching for personal navigation assistants", New Jersey TIDE center, 1996.

[8] C. Drane and C. Rizos, "Positioning Systems in Intelligent Transportation System", Artech House, 1998.

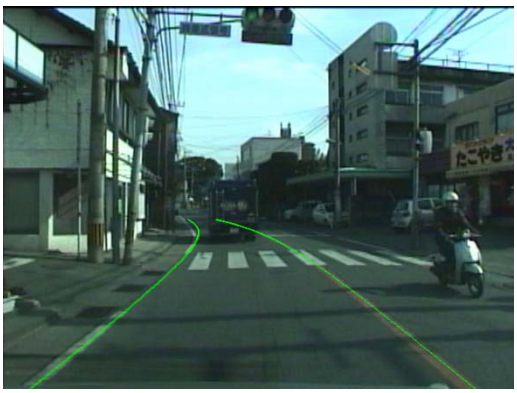
[9] S. Saab and Z. Kassas, "Map-Based Land Vehicle Navigation System with DGPS," ,Proc. of the 2002 IEEE Intelligent Vehicle Symposium, Versailles, France, pp. 209-214, June 2002

Table 1: Road parameters of figure 12 by proposed approach

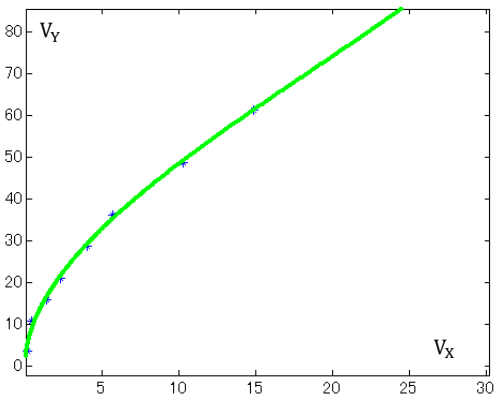
Fig	Width (m)	Offset (m)	Pitch (°)	Yaw (°)	CH_X (m^{-1})	CH_Y (m^{-1})
(a)(b)	3.33	1.60	1.28	1.50	-0.00826 -0.00312 1.65e-005	0.99946 -0.00027 1.43e-006
(c)(d)	3.24	1.40	0.00	0.00	0.00592 0.00517 -2.46e-005	1.0044 -0.00057 -6.62e-007
(e)(f)	3.25	1.80	0.77	-1.00	-0.0465 0.004 -4.10e-006	1.0007 4.892e-005 -7.92e-006



(a)

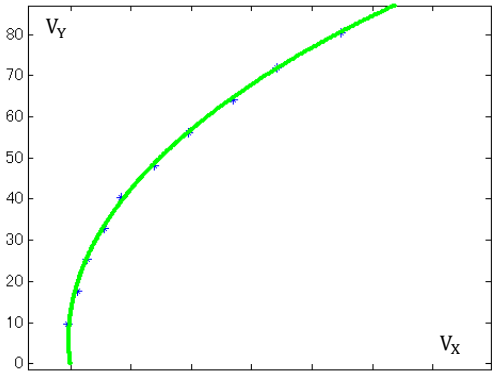


(b)



(c)

(d)



(e)

(f)

Fig. 12: Road geometry estimation and final detection result on image

Session III

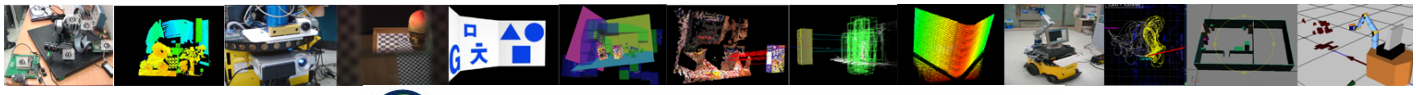
SLAM, Localization, Reconstruction

- **Title: Cognitive Localization of 3D Objects Symbolically Given Navigational Cues** *(invited paper)*
  Authors: Sukhan Lee, Hyunjun Kim, Zhaojin Lu, and Harry Hung

ICRA 2009 City Home Workshop - 2009

Object Recognition and Workspace Modeling for Cyber Transportation

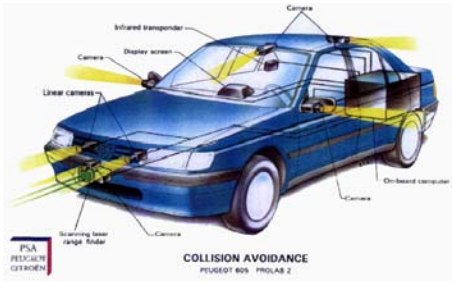
Sukhan Lee
Professor and Director



SUNG KYUN KWAN UNIVERSITY

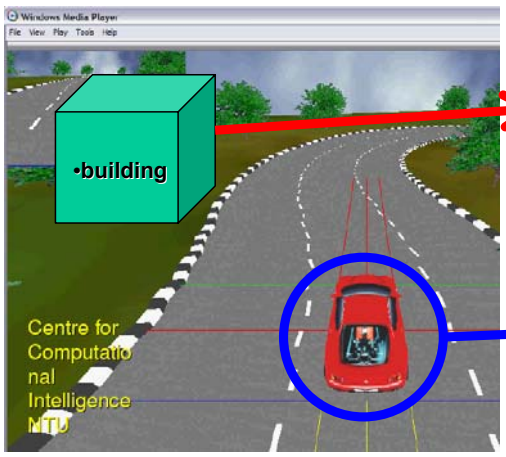
ISRC

Motivation



•Intelligent Vehicle

•Cyber Cab Service



•Building Recognition

•Localization

•Localization by Object Recognition

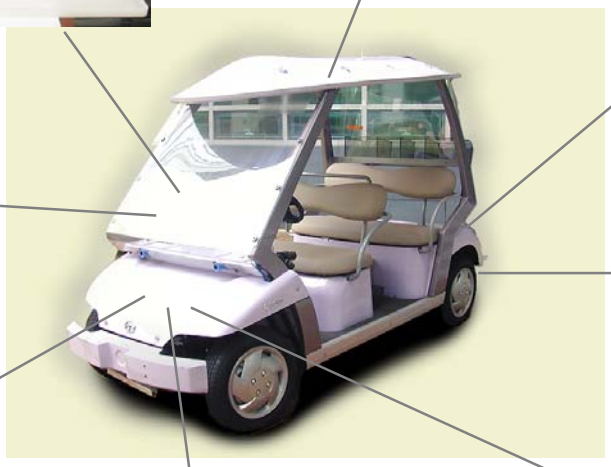
ISRC

Cyber car Experiment Platform

•Stereo Camera

•RearWheel Encoder

•Control Deck



•Main Control MCU ASS'Y

•RearWheel encoder Mount

•Front View

•Steering Encoder

•Terminal Box

ISRC

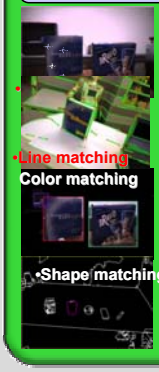
Overall Framework

•Object Recognition and Workspce Modeling for Cycab

Image Range Sensor

Data Acquisition

Feature Extraction



SIFT Feature

Line Feature

Color Feature

Boundary Feature

Object Recognition Framework

Particle Filter

SIFT Matching

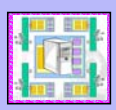
Line Matching

Focus of Attention

Evidence Selection

Evidence Collection

Knowledge Base System



Object DB

Environment DB

Human Info DB

Workspace Modeling

Global Geometry Extraction

Voxel Representation

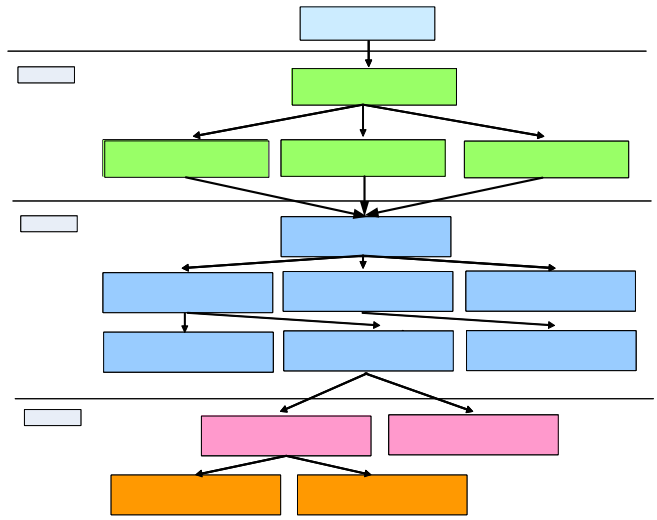
ISRC

Modeling based on generic knowledge in DB

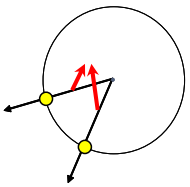
Real World

DB Update

•Object Modeling



< Generic Model >



$$N_1 = \begin{pmatrix} \frac{x_{p_1} - x_c}{r} \\ \frac{y_{p_1} - y_c}{r} \\ \frac{z_{p_1} - z_c}{r} \end{pmatrix} \quad N_2 = \begin{pmatrix} \frac{x_{p_2} - x_c}{r} \\ \frac{y_{p_2} - y_c}{r} \\ \frac{z_{p_2} - z_c}{r} \end{pmatrix}$$

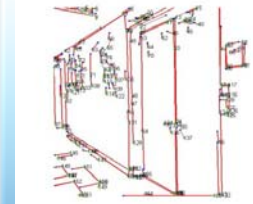
Cross-section and Center Point Equation
(Rotational Symmetry Modeling)

ISRC

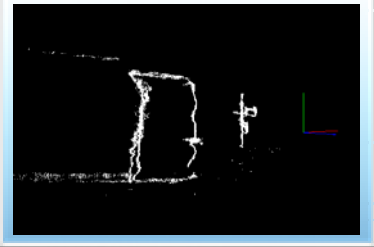
Parallel Line Model Matching

•Original Image

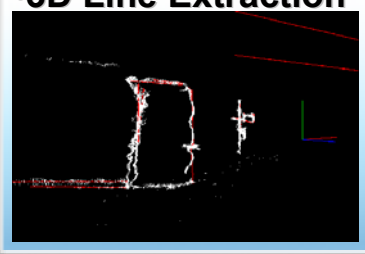
•2D Line Extraction



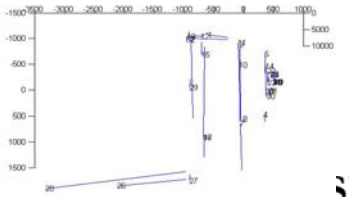
•3D Points Cloud



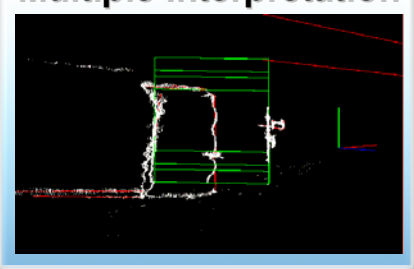
•3D Line Extraction



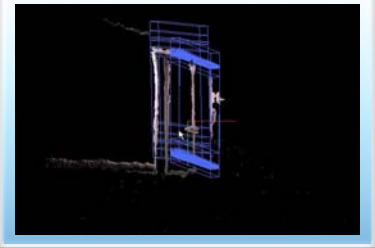
•Parallel Line Extraction



•Multiple Interpretation

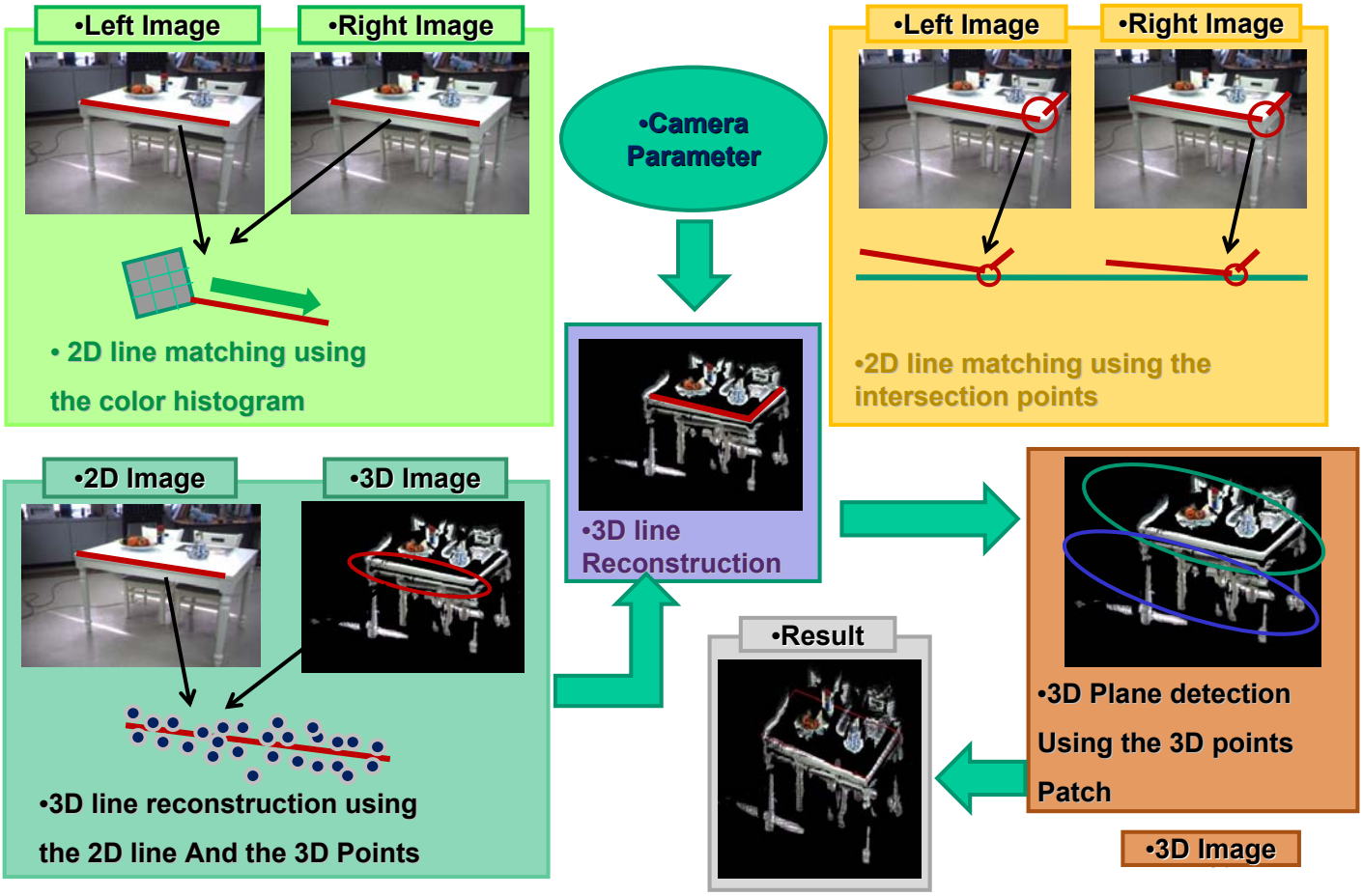


•Pose Estimation

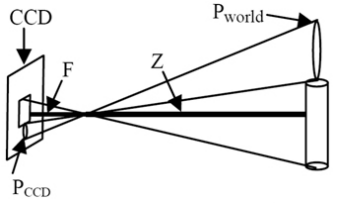


ISRC

Global Real Plane Detection



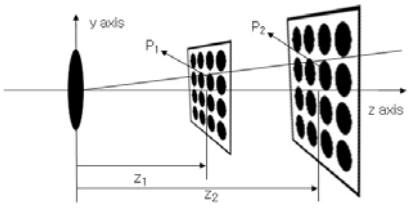
Octree Generation using Octree Fill-Factor



$$Fill\ Rate = \frac{Octs^2}{P_{CCD}{}^2 * Z / F}$$

$$P_{CCD} : P_{world} = F : Z$$

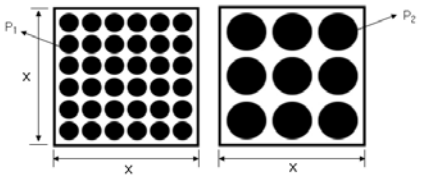
Octree Fill-Factor using Camera parameter



Using Distance Ratio between camera and a subject

$P_1 : P_2 = Z_1 : Z_2$

$P_2 = P_1 (Z_1 / Z_2)$



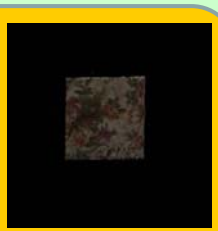
Getting the same points weight (a)and (b)

If z2/z1=α

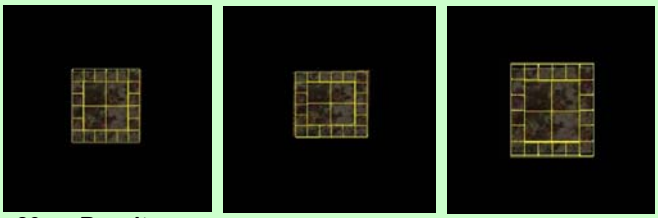
then P2=αP1

(a)' fill-rate ≒ (b)' fill-rate

The same size octree cell generation for the Same area of obstacle by the Fill-Factor

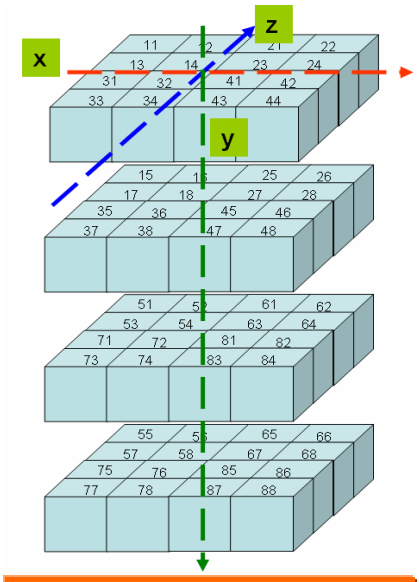


Experiment Image



80cm Result | 110cm Result | 140cm Result

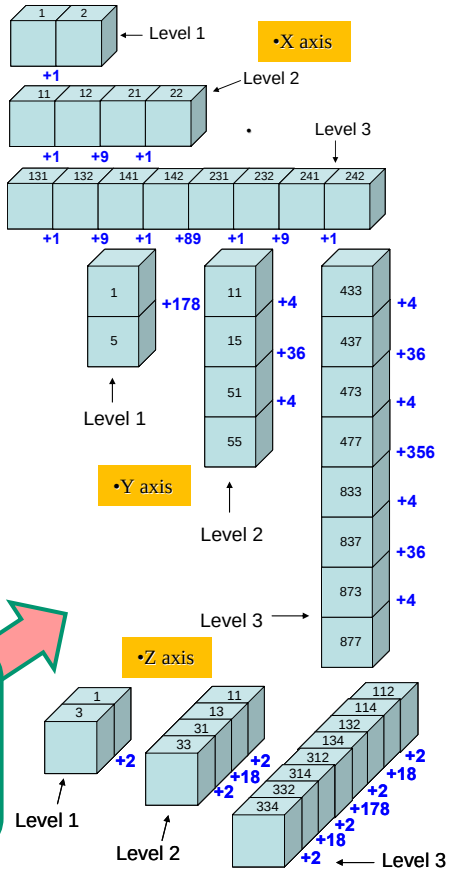
Octree Clustering For Multiple Octree Representation

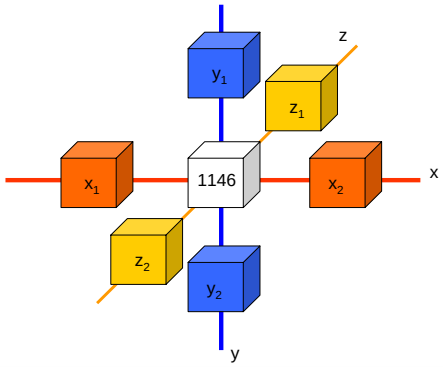


•Structure of the Octree Address

$$X\ axis = \{\pm 1, \pm 9, \pm 89, \pm 889, \pm 8889, ...\}$$
$$Y\ axis = \{\pm 2, \pm 18, \pm 178, \pm 1778, \pm 17778, ...\}$$
$$Z\ axis = \{\pm 4, \pm 36, \pm 356, \pm 3556, \pm 35556, ...\}$$

$$a_{n+1} = \pm(a_n + R_n)\ and\ R_n = 8 * a_0 * 10^n$$
$$(a_0 = initial\ value,\ 0 \le n \le \max length)$$





3 DIRECTION COMPLEMENT VALUES OF THE ADDRESS 1146

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
	COMP ADDR	COMP ADDR	COMP ADDR	COMP ADDR
X_1	-1	-9	-89	-889
	1145	1137	1057	257
X_2	1	9	89	889
	1147	1155	**1235**	2035
Y_1	-4	-36	-336	-3556
	1142	1110	810	-2410
Y_2	4	36	356	3556
	1150	**1182**	1502	4072
Z_1	-2	-18	-178	-1778
	1144	**1128**	968	-632
Z_2	2	18	178	1778
	1148	1164	1324	2924

COMP: COMPLEMENT VALUE, ADDR: ADDRESS OF OCTREE CELL, GRAY CELL IS THE FINALL ADDRESSING RESULT

•6 Neighbors address table

ISRC

Fuse Evidence by Particle Filter

$$p\left(\frac{X\ \&\ O}{m_1, m_2}\right) = p\left(\frac{X}{O, m_1, m_2}\right) \cdot p\left(\frac{O}{m_1, m_2}\right)$$

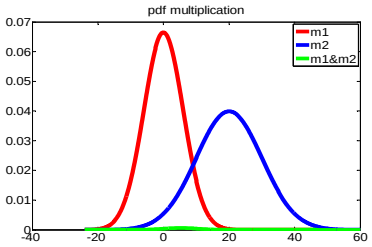
$p\left(\frac{O}{m_1, m_2}\right)$ •means that the probability of the recognized object given the interpr etations

$p\left(\frac{X}{O, m_1, m_2}\right)$ •means the pose distribution of the recognized object given the recognized results

$$p\left(\frac{O}{m_1, m_2}\right) = \frac{1}{1+\alpha}, \quad \text{where } \alpha = \frac{p\left(\frac{m_1, m_2}{\bar{O}}\right) p(\bar{O})}{p\left(\frac{m_1, m_2}{O}\right) p(O)}$$

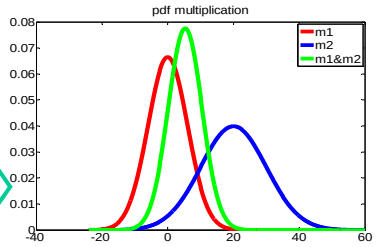
$$\alpha = \frac{p\left(\frac{m_1, m_2}{\bar{O}}\right) p(\bar{O})}{p\left(\frac{m_1, m_2}{O}\right) p(O)} = \frac{p\left(\frac{m_2}{\bar{O}}\right) p(\bar{O}) p\left(\frac{m_1}{\bar{O}}\right) p(\bar{O})}{p\left(\frac{m_2}{O}\right) p(O) p\left(\frac{m_1}{O}\right) p(O)} = \alpha_1 \alpha_2$$

•Given $p\left(\frac{O}{m_1}\right) = \frac{1}{1+\alpha_1}$ $p\left(\frac{O}{m_2}\right) = \frac{1}{1+\alpha_2}$



$$m \approx \frac{m_1\sigma_2^2 + m_2\sigma_1^2}{\sigma_2^2 + \sigma_1^2}$$
$$\sigma \approx \frac{\sigma_2^2\sigma_1^2}{\sigma_2^2 + \sigma_1^2}$$

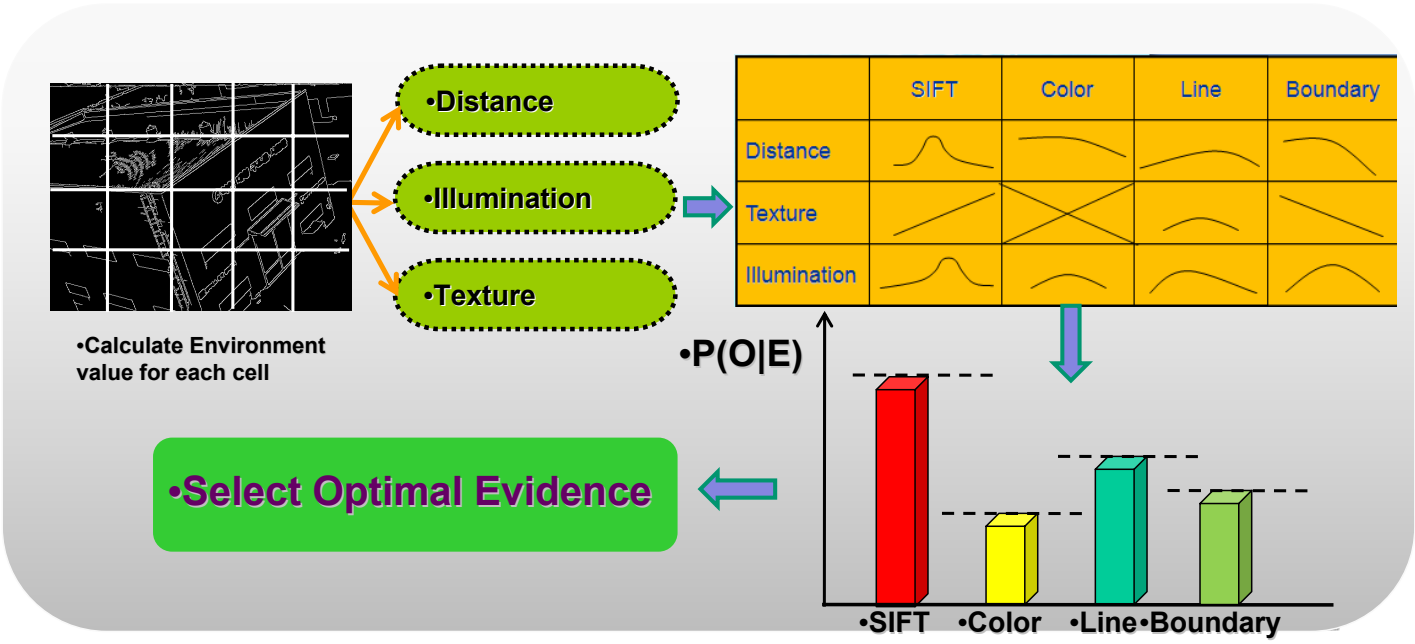
•After Normalization



ISRC

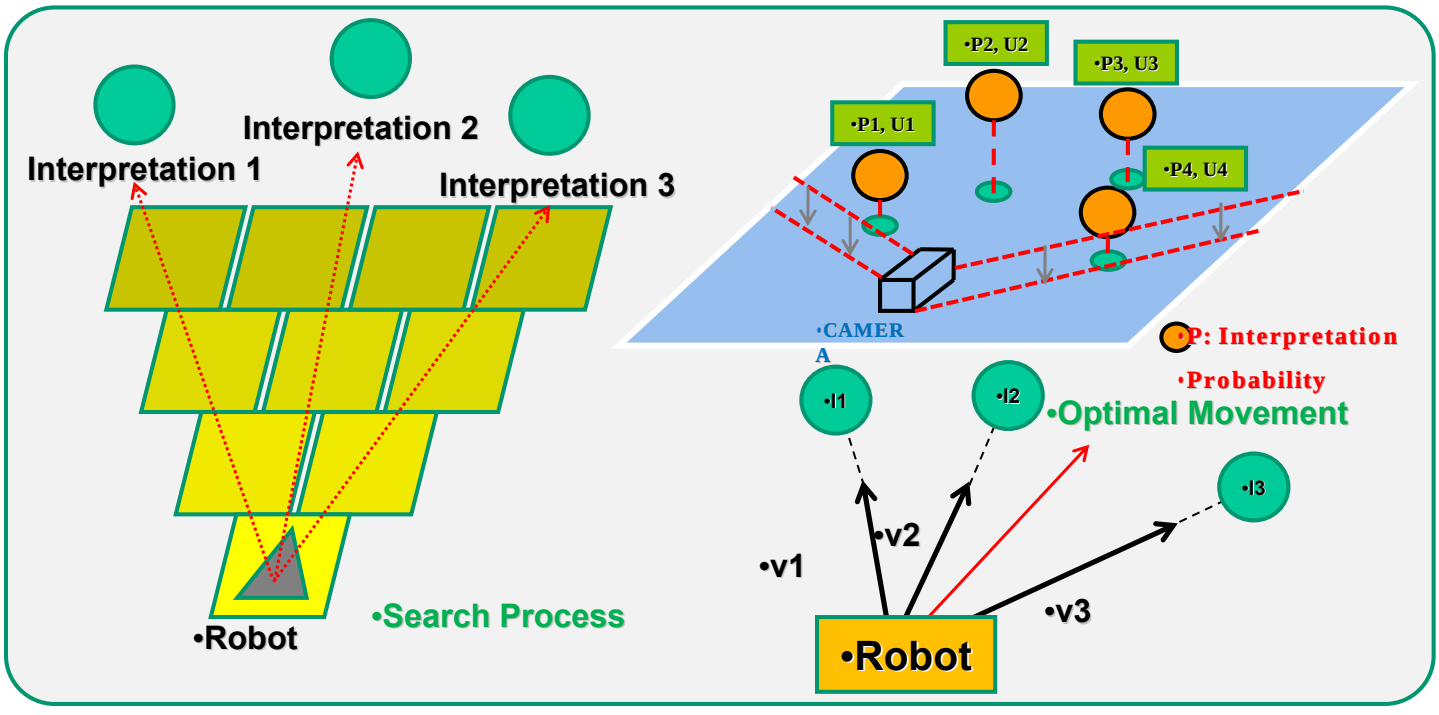
Focus of Attention & Evidence Selection

- Automatic perception of Environment change in ROI (distance, illumination, texture)
- Bayesian theorem based Optimal Evidence Selection
- Use Feature Evidence - Line, SIFT, Color & Boundary



Evidence Collection

- Robot Behavior for Minimize the Sensor Uncertainty & Entropy
  - Start Evidence Collection Process by Grid map based search process
  - Entropy Minimization Behavior by Project Multiple interpretations to 2D Map
  - Increase reliability of evidence by view change



Experiment & Results

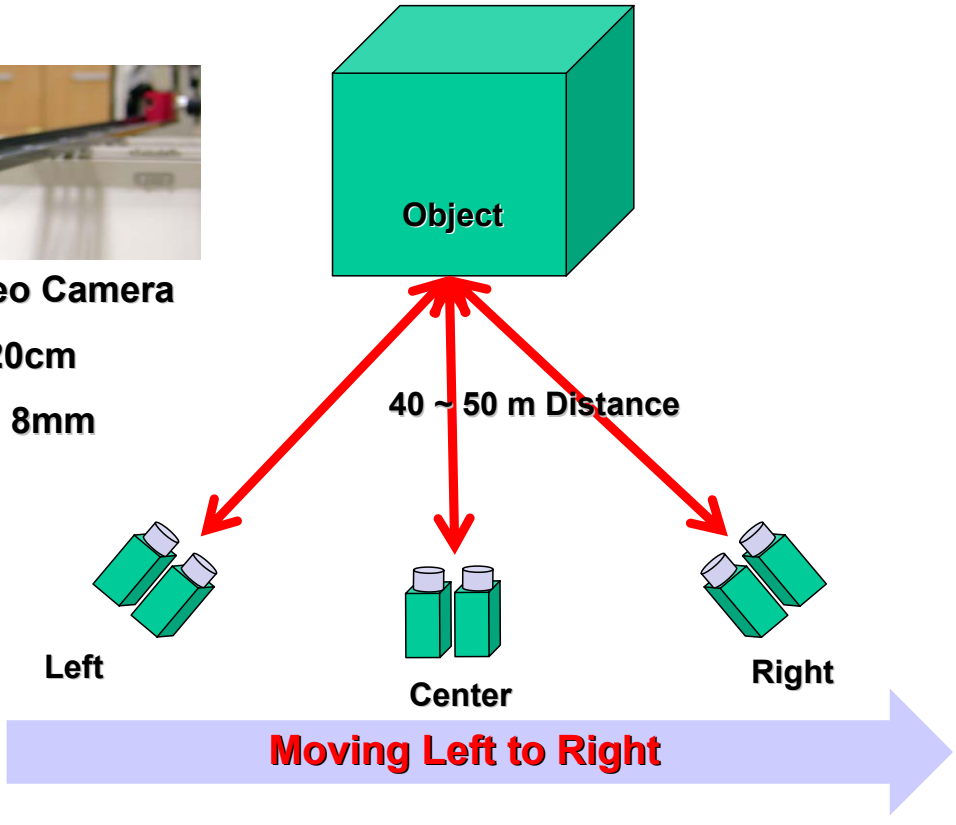
ISRC

Experiments

Wide Baseline Stereo Camera

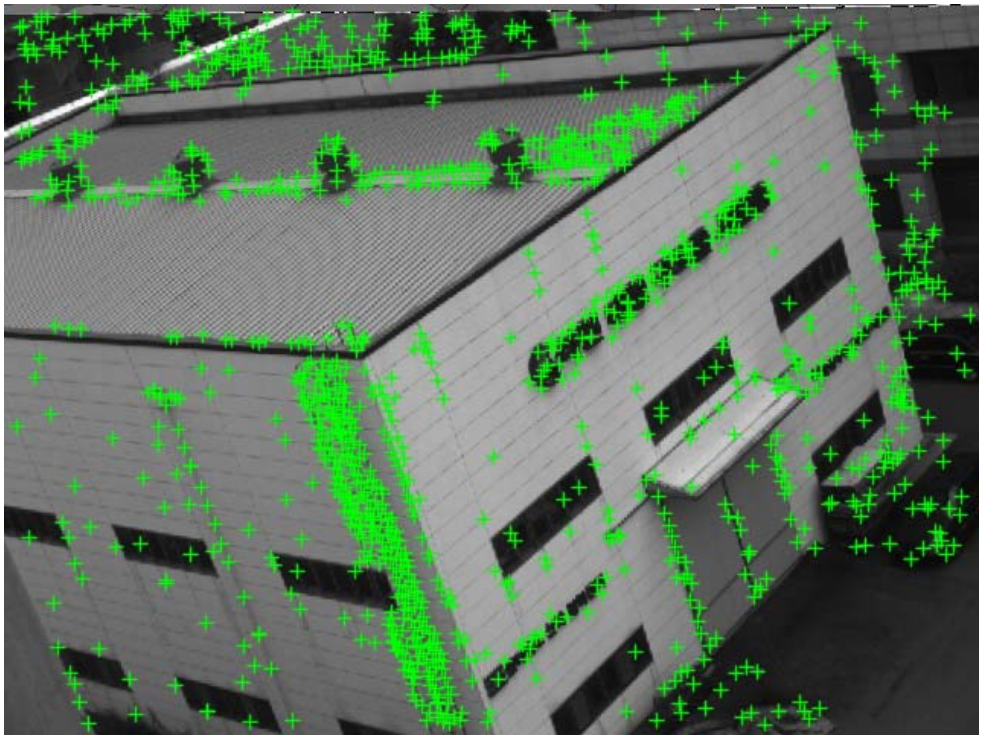
•Base line : 120cm

•Focal Length : 8mm



ISRC

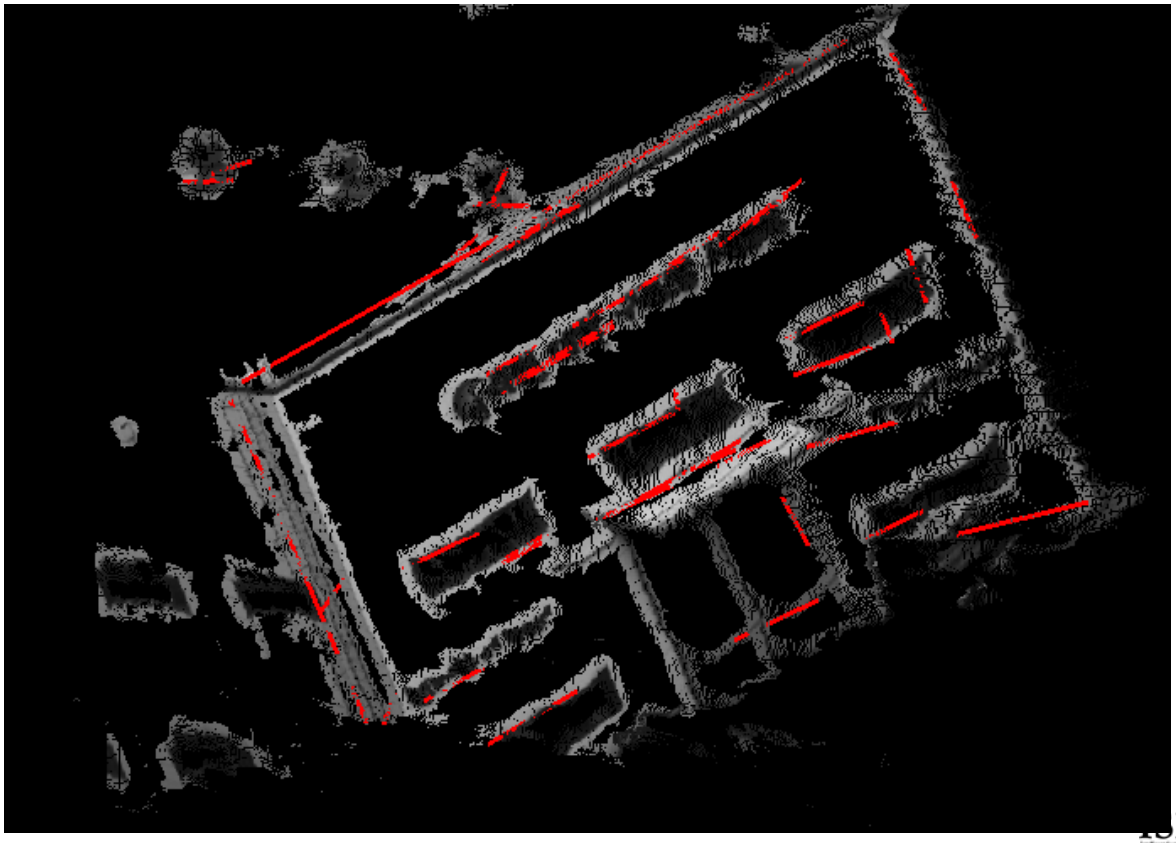
Result of SIFT Feature Extraction



ISRC

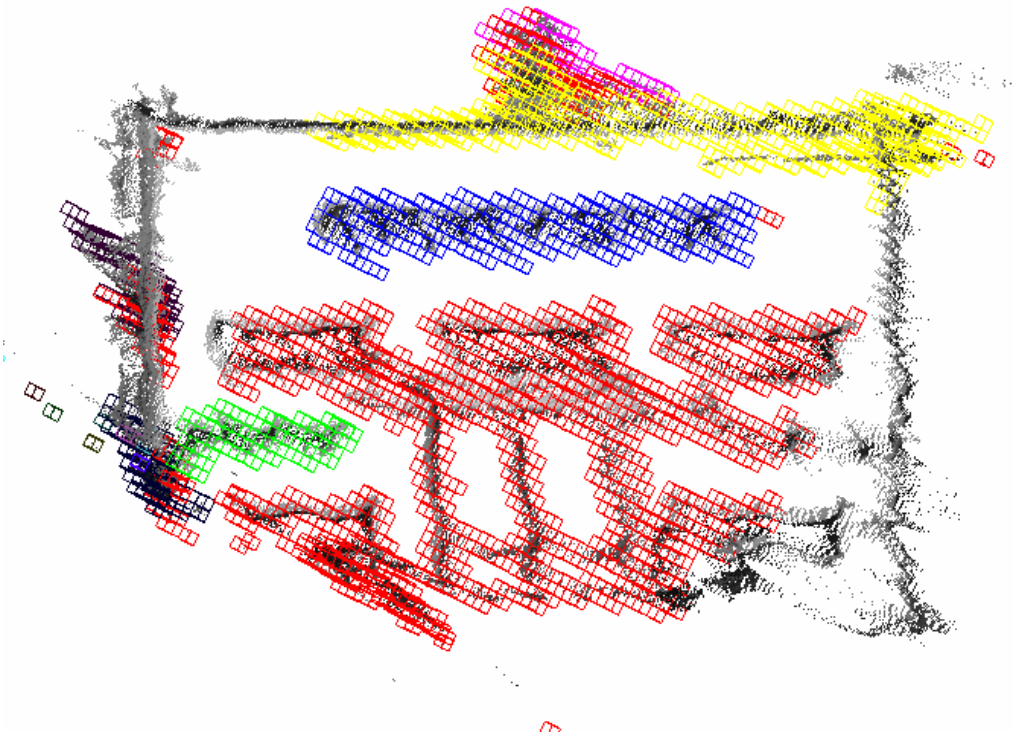
Result of 2D line detection

Result of 3D line detection



ISRC

Result of Octree Representation



ISRC

Experiments Results

Experiment A - Pose Estimation Result of the all evidences fusion (5 sequential scenes)

ISRC

Experiments Results (cont')

Experiment B - Pose Estimation Result of the all evidences fusion (10 sequential scenes)

ISRC

Conclusion

- Estimate different of SIFT result and 3D parallel line result
- Get more stable result by fuse of two result
- Average Processing Time per each scene
  - SIFT Matching : 0.8 sec
  - 3D parallel Line Matching : 1sec
  - Fusion & Filtering : 1.8sec
  - Centrino 2Hz CPU & 1G byte RAM
- Future works
  - Sensor fusion with GPS and wheel odometry information
  - To evaluate the pose estimation result

Thank You !

Session III

SLAM, Localization, Reconstruction

- **Title: Laser scanner based SLAM in real road and traffic environment**
  Authors: Olivier Garcia-Favrot and Michel Parent

LASER SCANNER BASED SLAM
IN REAL ROAD AND TRAFFIC ENVIRONMENT

Olivier Garcia-Favrot and Michel Parent
INRIA Paris - Rocquencourt
domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex France.

Abstract - In this paper we will present a SLAM algorithm we have recently developed for our needs in autonomous automotive applications. Our approach has the particularity of making use exclusively of laser scanners to achieve our goals without using any other type of sensors or source of information. We concentrated on developing a self-contained system that could be placed on any kind of mobile platform and work in any kind of dynamic environment; this is why too at this point our approach does not make use of any model of the vehicle. Our SLAM system has been tested with success both on a car at full speed on a road and a human evolving indoors. We will present here the challenges we face that pushed us to develop the algorithm, the solutions we are exploring, discuss experimental results and suggest areas of future work.

I - Introduction

When trying to make a vehicle autonomously travel to its destination over many kilometers you soon realize you will have to overcome many challenges [1],[2],[3],[4]. The first need is to be localized globally in order to know what path to take for reaching your destination. The second need is to be localized locally in the surrounding environment to make sure you are keeping the road, the right speed, the right distance with the vehicle in front, that you do not collide with other vehicles and obstacles etc. A third need is to have the ability to detect certain features in the environment in order to know how to properly behave on the road. For example, you need to be able to detect lanes in order to stay centered, you need to detect intersections, crosswalks, continuous or dashed lines etc… Then you have to take cost into consideration if you expect your work to have one day any use in real life. While global localization is a problem that can be considered to be solved thanks to a wide variety of affordable GPS solutions, all the other issues still present a big challenge when it comes to real world applications.

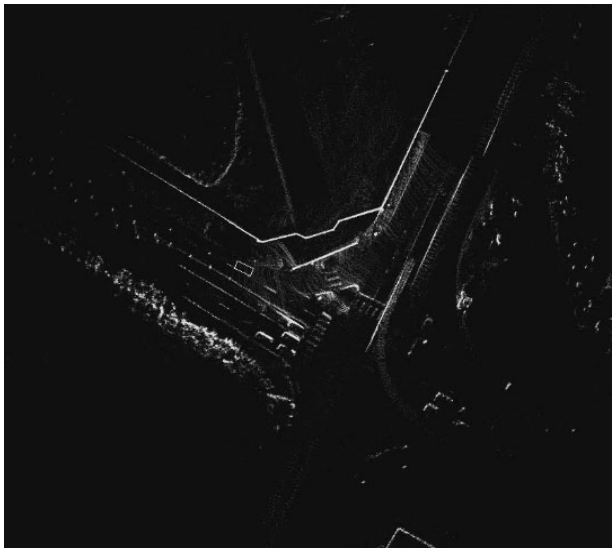


Fig 1. Map result showing road features.

In our struggle for achieving autonomous driving we have been exploring separately the potential of a single sensorial solution, the laser scanner, we believe eventually may give the necessary information to tackle all the issues mentioned above. We plan to use other types of sensors too but at this point the goal of our research is to make the most out of this type of sensor taken separately, before combining it with other sensorial input [5],[6].

Using a laser scanner only, we are able to keep track of our trajectory and our speed, in other words to localize, this, combined with the range data, allows us to create a model of our environment in the form of a map. This duality between creating consistent maps and localizing has been extensively studied as the Simultaneous Localization And Mapping (SLAM) problem [7],[8]. As we see in the figures presented (for example Fig 1), taken from real driving situations, the maps obtained contain all kind of useful information we plan to use for path planning and navigation, such as lane markings, intersections, crosswalks, empty parking space etc. But before entering into more details of our experimental results, we will first have an overview of the algorithm.

II - The algorithm overview

Our SLAM algorithm consists of two steps. A first step in which the relative movement of the vehicle is estimated and a second step in which this first estimation is refined by relocating the pose respect to the map we are progressively building. Figure 2 shows the different steps of the algorithm.

Pseudo code

```
void main ()
{
   //Init the pose and the robot's motion
   Pose = init_pose () ;
   Relative_motion = init_relative_motion () ;

   //Get the first scan and plot it on the map
   Scan = get_scan () ;
   Spikes = get_high_derivative_points (Scan) ;
   Previous_segments = calculate_segments (Spikes) ;
   Map = update_map (Pose,Scan, Map) ;

  //Localization loop
   while(true)
   {

    Scan = get_scan () ;
    Spikes = get_high_derivative_points (Scan) ;
    Current_segments  = calculate_segments (Spikes) ;
    Previous_motion = Relative_motion ;
    Relative_motion = calculate_motion (Previous_segments, Current_segments);
    if (Relative_motion = = NULL) Relative_motion = Previous_motion ;

    Pose_estimation = calculate_pose_estimation (Pose, Relative_motion) ;
    Previous_pose = Pose ;
    Pose = refine_pose_estimation (Pose_estimation, Scan, Map) ;
    Map = update_map (Pose,Scan, Map) ;
    Relative_motion  = recalculate_motion (Pose, Previous_pose) ;
    Previous_segments = Current_segments ;

   }//end of while
}//end of main
```

Fig 2. Steps of the algorithm.

II.1- Motion estimation: The first step works by tracking the motion between the current scan and the previous one. This is done by tracking points of high derivative, in other words by tracking the "spikes" that are apparent on the scan. This makes the algorithm absolutely universal as it will work in any structured or non structured environment. To find points of high derivative within the scan is easy but we need to identify them over the current scan and the previous one in order to discover the motion. What we do to perform this identification is to trace segments between the spikes and make use of invariants throughout the movement such as for example the length of those segments. For example, if we manage to find a segment of length "L" connecting two spikes in the current scan and we find also a segment of the same length in the previous scan, then we have pretty good chances that those two spikes at the ends of the segment are the same points of the environment. Now, by looking to our relative position to the segments in both the current and previous scans we are able to infer the relative translation and rotation we have done between the two scans. The problem is that many of the spikes are noise and are not consistent with the vehicle's movement. The challenge here was to develop the proper filter capable of sorting out only the spikes that are common to both the scans.

The typical error of the motion calculated this way is of ±20 cm in both x and y and ±2° in the orientation. Thankfully this error can be greatly reduced by refining this first estimation through a second step.

II.2- Relocation: The second step makes use of our near past experience by relocating on a map created by the previous scans. In our case a map is nothing else but a model, a discretization of the environment in the form of a bitmap containing a height field (Fig.3). The height field is formed as laser measurements are accumulated over time on certain pixels making the features of the environment become apparent. The amount of the laser data and its accuracy determine the level of discretization of the environment that can be achieved, that is, the map resolution (see section **II.3**). We believe that by studying the morphology of this height field we will be able to extract from it all the information necessary for navigation but this is a totally separate matter from localization.

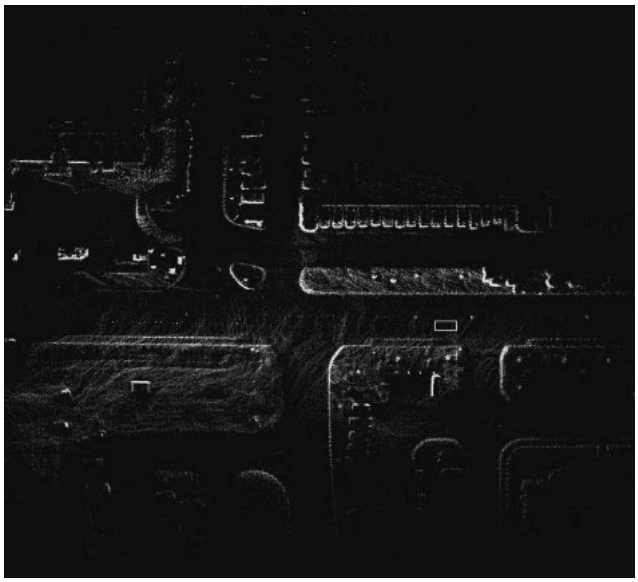


Fig 3. Bitmap containing the height field.

For relocation within the map, we use the fact that we know the actual pose of the vehicle is within the rather small error bounds of the pose estimation obtained from the first step. Since for relocation we are going to try to match the current scan with the map, the number of possible poses within the error bounds is limited by the resolution of the map. For example, if we have a 10cm per pixel map, the search area will be a square of 5x5 pixels and the discretization for the heading can be let say 0.2°. This gives us 25x21 = 525 candidate poses we then check exhaustively for the best match. We consider as the best match the candidate pose that maximizes the sum of the range value of the current scan points that hit non-empty pixels in the map. Of course, if our actual position is outside the typical bounds of error, then we are lost. For the rare cases when this happens we plan in the future to use more sophisticated techniques to find our position within the map but in practice this rarely happens except in the case the environment is very poor in information such as very large open spaces.

About this second step it is worth noting that the more information the map has about the zone of the current scan, the better it works. Although this seems evidence, it brings to an important conclusion which is that relocation works best when having a sensor looking in the opposite direction to the movement. This is so because unless we have done a loop, in general we are exploring what lies in front while we have already explored what lies behind us and therefore we have much more information. This is especially true at high speeds, looking at Figure 4 we can see the height field is much clearer in the opposite direction of the movement as we have gathered much more information in the zone of the map we have left behind than the amount we have managed to gather of what is coming in front of us. This means that simply having a sensor in the front of the vehicle is conceptually a bad policy if we plan to do localization, especially at high speeds. Also, in an automotive application it is of utmost importance to include the rear of the vehicle in the field of view as otherwise it would be like driving without any mirrors.

Finally, once we have refined our location within the map, we update the height field with the current scan data and we recalculate at posteriori the relative movement we have done.

It may happen to the first part of the algorithm not to give any output, in such cases we simply fill the gap taking as our current relative movement the last one we were able to calculate.

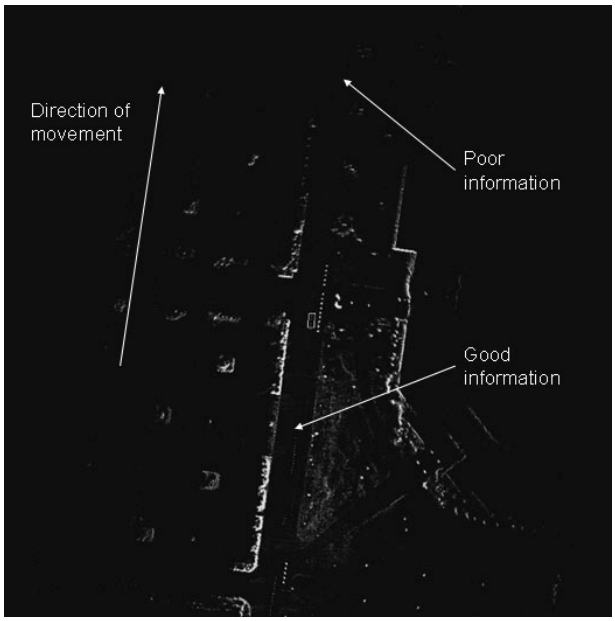


Fig 4. Best info is found at the back of the vehicle.

II.3- Algorithm's behavior: Although there is certainly a cumulative error out of this two step algorithm, it is so small that for practical purposes there is none. As we explained in the introduction it is not our goal to make use of this localization over hundreds of kilometers, this task is left to the GPS. Our use of the algorithm is to be localized locally, that is, to be able to generate an accurate map of the surrounding over 100 m of our vehicle in order to extract from it information about lanes, obstacles and other vehicles and take it as a base for path planning and trajectory control.

Even though, we have to say that our experimental results over few kilometers show very little accumulated error (see section III for more details). Errors mainly seem to occur as a result of singularities, such as having a surrounding environment very poor in information, that is, when we happen to go through large open spaces. This is to be expected as the accuracy and quality of the output of the algorithm varies depending on the quality and amount of the information you feed in it.

A study on the subject has not been realized yet at this point but globally we have the following:

The map resolution that can be achieved in general is very dependent on the angular resolution and the scan frequency. Without surprise the scan frequency is particularly important at high speeds. Other parameters like the laser range and field of view have no remarkable effect on the achievable map resolution. Actual tested implementations with different settings gave a 10 cm per pixel map resolution at 10 Hz and 0.5° angular resolution and a 5 cm per pixel map resolution at 20 Hz and 0.25° angular resolution.

When it comes to loss of localization, as we have explained, open spaces are our major enemy. It is therefore no surprise that the robustness of the algorithm is directly proportional to the range of the laser and how wide is our field of view, as this helps to reduce greatly the cases in which the algorithm has nothing to "hook" on.

Another potential source of error is moving objects. The algorithm normally filters out the segments including moving points because they do not maintain the invariants found in segments from static points. This works only if the scans contain a sufficient number of static points. The field of view is therefore quiet important for dealing with this. A 270° field of view is generally enough for completely removing this issue as a source of error even in very heavy traffic conditions.

III - Real road testing

In order to test our algorithm in a real driving situation we went to see our partners at the Southwest Research Institute in San Antonio, Texas. We have to thank the whole SSTI team for their help and for making this possible.

The SSTI vehicle (Fig 5) is a fully automated Ford Explorer equipped between other things with two Alasca Ibeo laser scanners and a differential omnistar GPS we will only use for reference. Each of the Alasca scanners have a range of up to 200 m, have 4 layers and a 270° field of view. They are located at the left and right front corners of the vehicle, so together they cover almost 360° except for the very rear part of the vehicle.

Fig 5. The SSTI vehicle at the SwRI in San Antonio.

Alasca laser scanners have both positive and negative specifications when it comes to our algorithm. On one side they have a long range, a 270° field of view and 4 simultaneous scanning planes at different angles (Fig 6). In the end we make use of only one of the planes in the localization process but the information of the four combined makes possible to obtain very rich and detailed maps. On the other side, each separate scanning plane has a very poor angular resolution which only gets worse when increasing the scanning frequency.

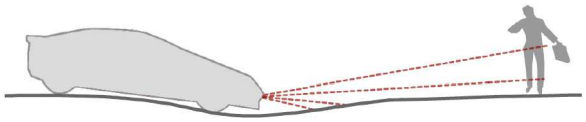


Fig 6. The laser scanners have 4 simultaneous layers.

Although the different scanning planes are interlaced and we have tried to combine them, this does not work because of the fact the points of high derivative we use keep information about the movement only when you derivate a continuous signal coming from the same scan plane. To make things worse, by default the angular resolution is not constant but higher in the front and lower on the sides and rear, which as we explained earlier is the best zone for relocation. As a result of this, the implementation on the SSTI vehicle was limited to achieve a 20 cm per pixel map resolution but turned out to be very robust thanks to the almost 360° field of view and possibly to its low map resolution. The scanning frequency was set to 12.5 Hz which may not be sufficient when driving at high speed but to increase the frequency would have lowered too much the angular resolution.

On the whole, results were very satisfactory as we were able to localize over kilometer drives at high speeds (60km/h to 70km/h) on a real road and heavy traffic environment.

Because the four scanning planes were plotted, rich and detailed maps were obtained with road features becoming apparent such as continuous and dashed lane markings (Fig 7). We believe this happens because as the lower scans sweep the road at a certain distance and angle, only reflective surfaces send back an echo. At this point we don't take advantage of this for localization because the lower scans are not used in the process.

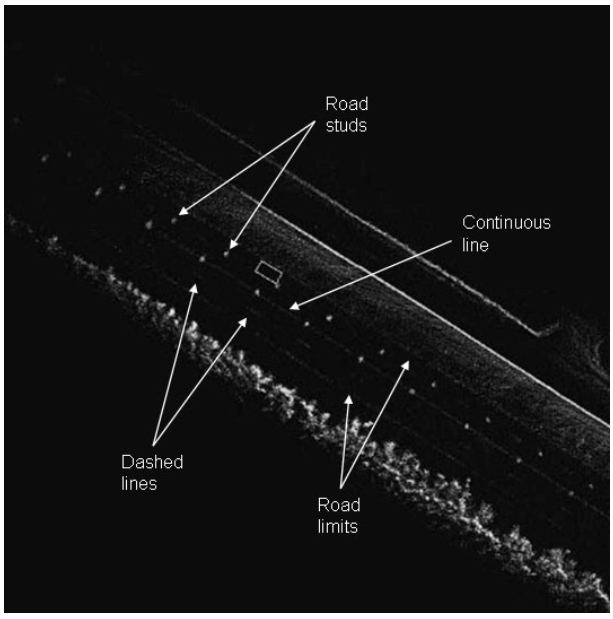


Fig 7. Road features become apparent.

We have to point out that when driving over large distances the use of a static map is not practical. This is why for the real road tests we have used a dynamic map centered on the vehicle and which we build from a circular buffer containing the N previous scans, that is, when new data comes in, the oldest data in the buffer disappears. The result covers a square area of 200m x 200m around the vehicle (1000 pixels x 1000 pixels) and we say the map is dynamic because it is being completely redrawn at each new scan. A dynamic map of the sort has been preferred during the tests to a static one primarily because the large distances involved in the testing would have required a huge bitmap. Then this is ok because in this application the map information is normally used for local navigation, which generally only involves a few meters around the vehicle. Other benefits of a dynamic map is its ability to regenerate rapidly in the case of a loss of localization and if a small buffer is used, the capacity of reflecting the changes in the environment. For example an opening gate that was initially closed will disappear from the map as it opens. Another example would be to update empty spaces in a parking lot while searching a place to park.

In the case we are dealing with an application in which we want to have a map over the complete trajectory of the vehicle, we need as we go to keep record of all measurements along with the calculated associated position. For drawing a portion of a map we then need to specify a location and we search in the recorded data all the positions found to be within a certain radius of the selected location. Now, we obtain a map by simply applying the appropriate coordinate change to the associated scans of those found positions. Because the map is kept not as a bitmap but reconstructed from the raw scan data every time, it is possible to easily zoom in or zoom out or rotate the map at will. This can be useful for example if we want the orientation of the map to follow in real time the current orientation of the vehicle.

Even though it is not the purpose of our algorithm, we are going to show here an example of its accuracy over a long distance drive (2.18Km). During the test, the differential GPS position was recorded to be used as a reference.

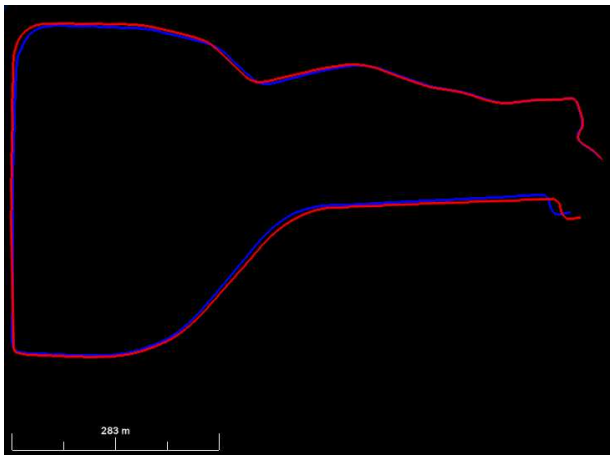


Fig 8. Laser(blue) and GPS(red) paths (2.18Km)

On this ride (Fig 8.), the differential GPS and the laser trajectories seem to match relatively well the first 400m but after an offset appears that remains for the rest of the way. Even though, comparing the relative shape of the trajectories and the distance involved, we can see the algorithm in general performs quite well even if it is almost inevitable to have at some moment some surrounding environment that the algorithm won't be capable of dealing with. We want to point that what we see here is the result of the raw output of the localization process, it could be certainly improved for example adding a Kalman filter which points towards interesting future developments.

IV - Suggested areas of future work

Immediate planned future work includes testing the algorithm with other laser configurations to see if we can improve its performance both in map definition and robustness. After that we plan to do:

- Tracking of moving objects, this should improve the localization process as we can eliminate the moving points from the next iteration of step one as well as eliminate those from the map, improving in this way the relocation in step two.

- Elaborate a good control law working with the map localization.

- Create classifiers able to extract from the map features such as lanes, intersections and so on.

- Combine the algorithm output with a low cost 1Hz GPS in order to transform the map coordinates into global latitude and longitude coordinates for global navigation.

REFERENCES

[1] S.Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aaron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "*Stanley: The robot that won the darpa grand challenge*," Journal of Field Robotics, vol. 23, no. 9, pp. 661-692, September 2006.

[2] C. Laugier, S. Petti, D. A. Vasquez Govea, M. Yguel, T. Fraichard, and O. Aycard, "*Steps towards safe navigation in open and dynamic environments*," in Autonomous Navigation in Dynamic Environments: Models and Algorithms, ser. Springer Tracts in Advanced Robotics Series (STAR), C. Laugier and R. Chatila, Eds. Springer, 2006.

[3] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "*Toward urban driverless vehicles*", International Journal of Vehicle Autonomous Systems, Special Issue on Advances in Autonomous Vehicle Technologies for Urban Environment, vol. 1, no. 6, pp. 4 – 23, 2008.

[4] S. Kolski, K. Macek, L. Spinello,and R. Siegwart, "*Secure autonomous driving in dynamic environments: From object detection to safe driving.*" in Proceedings of the Workshop on Safe Navigation in Open and Dynamic Environments: Applications to Autonomous Vehicles at the IEEE International Conference on Robotics and Systems, San Diego, USA, 2007.

[5] Lars B. Cremean and Richard M.Murray, "*Uncertainty-based Sensor Fusion of Range Data for Real-time Digital Elevation Mapping (RTDEM).*" submitted, 2005 IEEE International Conference on Robotics and Automation.

[6] L. Conde Bento, Urbano Nunes, Fernando Moita and Antonio Surrecio, "*Sensor Fusion for Precise Autonomous Vehicle Navigation in Outdoor Semi-structured Environments.*" Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems Vienna, Austria, September 13-16, 2005.

[7] S.Thrun. "*Robotic mapping: A survey.*" in G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.

[8] Bailey, I. *Mobile robot localisation and maping in extensive outdoor environments*. PhD thesis, ACFR, University of Sydney, Australia, Aug 2002.

Session IV

Motion planning

- **Title: Fast and Feasible Deliberative Motion Planner for Dynamic Environments** *(invited paper)*
  Authors: Mihail Pivtoraiko and Alonzo Kelly

- **Title: ICS-AVOID, a Collision Avoidance Scheme for Dynamic Environments**
  Authors: Luis Martinez-Gomez and Thierry Fraichard

- **Title: Mapping Obstacles to Collision States for On-line Motion Planning in Dynamic Environments**
  Authors: Oren Gal and Zvi Shiller

- **Title: Probabilistic Rapidly-exploring Random Trees for autonomous navigation among moving pedestrians**
  Authors: Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier

Session IV

Motion planning

- **Title: Fast and Feasible Deliberative Motion Planner for Dynamic Environments**
  (invited paper)
  Authors: Mihail Pivtoraiko and Alonzo Kelly

Fast and Feasible Deliberative Motion Planner for Dynamic Environments

Mihail Pivtoraiko and Alonzo Kelly

Abstract— **We present an approach to the problem of differentially constrained mobile robot motion planning in arbitrary time-varying cost fields. We construct a special search space which is ideally suited to the requirements of dynamic environments including a) feasible motion plans that satisfy differential constraints, b) efficient plan repair at high update rates, and c) deliberative goal-directed behavior on scales well beyond the effective range of perception sensors. The search space contains edges which adapt to the state sampling resolution yet aquire states exactly in order to permit the use of the dynamic programming principle without introducing infeasibility. It is a symmetric lattice based on a repeating unit of controls which permits off-line computation of the planner heuristic, motion simulation, and the swept volumes associated with each motion. For added planning efficiency, the search space features fine resolution near the vehicle and reduced resolution far away. Furthermore, its topology is updated in real-time as the vehicle moves in such a way that the underlying motion planner processes changing topology as an equivalent change in the dynamic environment. The planner was originally developed to cope with the reduced computation available on the Mars rovers. Experimental results with research prototype rovers demonstrate that the planner allows us to exploit the entire envelope of vehicle maneuverability in rough terrain, while featuring real-time performance.**

I. INTRODUCTION

Capable motion planners are important for enabling field robots to perform reliably, efficiently and intelligently. Despite decades of significant research effort, today the majority of field robots still exhibit various failure modes due to motion planning deficiencies. These failure modes range from computational inefficiencies to frequent resort to operator involvement when the autonomous system takes unnecessary risks or fails to make adequate progress. Based on our field robotics experience, we have developed a motion planning method that addresses many drawbacks of leading approaches. It is a deterministic, sampling-based method. It features a sampling of robot state space that lends itself well to utilizing standard graph search techniques, while enabling an array of high-performance planning capabilities. The proposed motion planning method was implemented and successfully validated in field experiments at the California Institute of Technology, Jet Propulsion Laboratory (JPL; Figure 1).

While the planner was originally developed for the Mars rovers, it is equally relevant to dynamic environments. In fact, a variant of this planner was used in CMU's winning entry in the DARPA Urban Challenge [1]. Motion planners intended for use in dynamic environments must be:

- Fast, in order to replan on-the-move when changes in the environment are detected,
- Feasible (meaning satisfying differential constraints) at least in the near field, so that the predicted path to avoid dynamic obstacles is the one actually executed by the vehicle,
- Deliberative, so that effective goal-directed behavior can be maintained despite the path perturbations caused by reacting to changes in the environment such as the motions of dynamic obstacles and the appearance and disappearance of local minima.

In order to satisfy the differential constraints, we propose to encode them in the search space. This allows to shift the constraint satisfaction to the search space design process, which can be accomplished *a priori* and off-line. In particular, we enforce that the edges of the graph that represents the planning problem represent the feasible motions that can be directly executed by the robot. In this manner, the on-line planner can utilize unconstrained, standard search algorithm to find a solution to the motion planning problem, a path in the representation graph.

For most systems featuring differential constraints, relatively high dimensionality of the state space may be required. Deterministic search in this setting can be computationally costly. Planning in complex outdoor environmens can be especially computationally expensive due to any combination of scale, dynamics, and dense obstacles.

Fig. 1. FIDO rover navigates autonomously using the proposed motion planner among dense obstacles in the Mars Yard at the California Institute of Technology, Jet Propulsion Laboratory. The planner manages frequent updates of the limited perception system by replanning continuously at approximately 10Hz. Computed planner motions are smooth in curvature and executable by the robot verbatim.

This research was conducted at the Robotics Institute of Carnegie Mellon University, sponsored by NASA/JPL as part of the Mars Technology Program.

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

This paper proposes a two-fold solution to this difficulty. First, the search space is designed to be compatible with replanning algorithms [2], [3] that can repair the motion plan efficiently when the representation of the environment changes. This is particularly important in robotics applications in partially known or dynamic environments, where changes in environment information are frequent. The robot must be able to recompute its motion plan quickly. The second manner of alleviating computational complexity of planning is through modification of the fidelity of representation of the motion planning problem. The proposed search space consists of one or more arbitrary regions of different fidelities. Lower fidelity of representation results in faster search, but higher fidelity results in better quality solutions. The approach is closely related to multi-resolution planning [4], but we use the term *graduated fidelity* to emphasize that the quality of representation is expressed not only as the resolution of state discretization, but also as the connectivity of edges between the vertices in the state lattice. Each region of the search space can be assigned a fidelity arbitrarily, yet practically this choice is guided by the region's relevance for the planning problem and the availability of the environment information. In particular, it is often beneficial to utilize a high fidelity of representation in the immediate vicinity of the moving robot. Our method meets that need by allowing the regions of different fidelity to move or change shape arbitrarily.

The contribution of this work is a representation of motion planning problems under differential constraints that has a number of important advantages:

- Satisfaction of differential constraints is accomplished off-line, to allow fast on-line perfomance,
- Compatibility with standard replanning algorithms allows quick robot reaction to changing environment information,
- Fast planning is enabled through managing the fidelity of representation.

Our concentration on achieving a fast deterministic constraint-compliant planner was originally motivated by the spartan processing available on the Mars rovers. However, our results are equally applicable to terrestrial problems where high planner update rates can be used to respond effectively to changes in dynamic environments as they are predicted or discovered by perception.

In the next section, we relate the proposed approach to prior work. In the following three sections, we further describe each of its principal benefits, as listed above. We will conclude this presentation with experimental results.

II. PRIOR WORK

A planner based on A* search in the state lattice was successfully utilized to guide a car-like robot in challenging natural environments [5]. The graduated fidelity extension to the state lattice is related to the general area of multi-resolution planning: [6], [7], [8] and others. One difference our method has with most multi-resolution predecessors is that we allow regions of different resolution to move over time, while the search space remains compatible with systematic search. The idea of dividing the search space into regions is related to [9], but our method allows replanning in this search space.

Satisfaction of differential constraints also has received a considerable amount of attention in motion planning research. Powerful probabilistic techniques have been developed [10] [11], however our method is deterministic and under appropriate conditions can offer a number of guarantees provided by standard search algorithms, including optimality and resolution-completeness. A number of other approaches utilize discretization in control space to manage the complexity of the planning problem [12]. However, there are important advantages to using discretization in the state space instead. In particular, it simplifies reducing dispersion of sampling, in turn allowing a more uniform distribution of samples in the state space [4]. This is beneficial to exploring the state space more efficiently, as the search attempts to find a path from initial to final state. Unfortunately, reducing state space dispersion through control space sampling is difficult. It was shown in [16] that through careful discretization in control space, it is possible to force the resulting reachability graph of a large class of nonholonomic systems to be a lattice, however this is usually difficult to achieve. By using a boundary value problem solver [14], we are able to choose a convenient discretization in the state space, one that makes the search more efficient, while maintaining the satisfaction of differential constraints.

III. FEASIBLE MOTIONS

Discrete representation of robot state is a well-established method of reducing the computational complexity of motion planning. This reduction comes at the expense of sacrificing *feasibility* and *optimality*, the notions denoting the planner's capacity to compute a motion that satisfies given constraints, and to minimize the cost of the motion, respectively. In computing motions, we seek to satisfy two types of constraints: features of the environment that limit the robot's motion (*obstacles*) and the limitation of the robot's mobility due to the constrained dynamics of its motion (*differential constraints*). Motions that satisfy both types of constraints will be referred to as *feasible* motions.

The proposed method is based on a particular discretization of robot state space, the *state lattice*. It encodes a graph, whose vertices are a discretized set of all reachable states of the system, and whose edges are feasible motions, *controls*, which connect these states exactly. The motions encoded in the edges of the state lattice form a repeating unit that can be copied to every vertex, while preserving the property that each edge joins neighboring vertices exactly. This property of the search space will be denoted *regularity*. The canonical set of repeating edges will be called the *control set*. The number of edges in the control set is exactly the branching factor, *out-degree*, of each vertex in the reachability graph.

A. Sampling State and Motions

Beneficial state sampling policies include regular lattice sampling, where a larger volume of the state space is covered with fewer samples, while minimizing the dispersion or discrepancy [4]. It is natural to extend the concept of regular sampling from individual values of state to sequences of states (i.e. paths). As for state space, the function continuum of feasible motions can also be sampled to make computation tractable. The effective lattice state space sampling, developed in this work, induces a related effective sampling of motions.

Suppose discrete states are arranged in a regular pattern. Besides sampling efficiency benefits, an important advantage of regular sampling of state space is (quantized) translational invariance. Any motion which joins two given states will also join all other pairs of identically arranged states. By extension, the same set of controls emanating from a given state can be applied at every other instance of the repeating unit. Therefore, in this regular lattice arrangement, the information encoding the connectivity of the search space (ignoring obstacles) can be pre-computed, and it can be stored compactly in terms of a canonical set of repeated primitive motions, the control set. Two properties of lattice search spaces that are necessary conditions for satisfying differential constraints are:

1) Enforcing continuity of relevant robot state variables across the vertices,
2) Ensuring that the edges between the vertices of the search space represent feasible motions.

The first condition can be satisfied by adding the relevant dimensions to the search space, in order to represent the continuity of state variables explicitly. For example, if a heading dimension is added to the a 2D (x,y) state space, then $(x, y, North)$ and $(x, y, East)$ become distinct states. In order to satisfy the second condition, we require a method of discretizing the robot control space to force its reachability tree to be a regular lattice in state space. We identify two methods of achieving this:

- *Forward* — for certain systems, there are methods of sampling the control space that result in a state lattice [16], [15],
- *Inverse* — a desired state sampling can be chosen first, and boundary value problem solvers can be used to find the feasible motions (steering functions) that drive the system from one state value to another, e.g. [14], [17].

We prefer the inverse approach because it permits the choice of state discretization to be driven by the application – including the vehicle and the environment. Smaller state spacing is desired for denser obstacles or smaller vehicles. Note that, in the state lattice, if state separations are small relative to the distance required to change vehicle heading by the distance to the next heading sample, the edges in such a structure can span many state separations.

Fortunately, the work of constructing the state lattice can be performed off-line, without affecting planner runtime. Once it is constructed and represented as a directed graph (compactly specified with a control set), the state lattice can be searched with standard algorithms. An example of a simplified state lattice is shown in Figure 2.

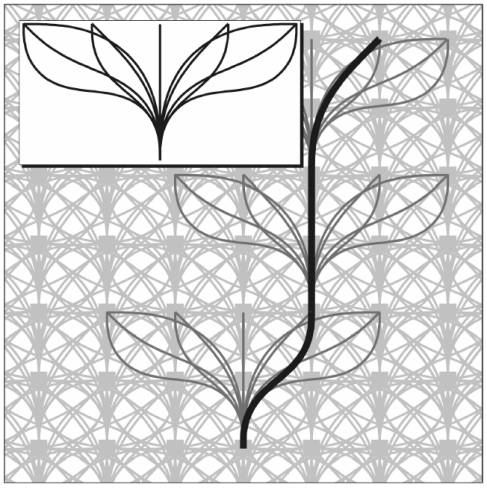


Fig. 2. An Example State Lattice. A repeated and regular pattern of vertices and edges comprises the state lattice. The inset shows the control set, the motions leading to some nearby neighbors of a vertex. The overall motion plan (thick black curve) is simply a sequence of such edges. Reverse motions were omitted for clarity.

Algorithm 1 is a simple inverse method for generating a control set. Referred to as the Shortest Edges algorithm, it may serve as a departing point to evaluate our proposed approach to search space design. To better illustrate the algorithm, in this section we assume a 4D state lattice that consists of 2D translation, heading and curvature. Suppose that Θ and K are user-defined subsets of discrete values of heading and curvature in the state lattice, respectively. By exploiting rotational symmetries in the state lattice, these sets can be desired strict subsets of all possible discrete values of these states variables. The outer for-loop selects the permutations of discrete values of initial and final heading and curvature. The inner for-loop cycles through all discrete value pairs of x and y, such that the maximum norm[1] L_∞ between the origin O and (x_f, y_f) grows from 1 to infinity. For each value of L_∞, if the trajectory generator finds a solution to the boundary value problem, a feasible trajectory u_i, we add it to the control set. At this point we break from the inner for-loop and proceed with another choice of terminal heading and curvature values. The algorithm terminates when a trajectory is generated for every permutation of heading and curvature values.

[1] L_∞ norm of a vector $\mathbf{x} = [x_1, x_2, ..., x_n]$ is $\max_i |x_i|$

Input: State discretization in the state lattice: position, discrete values of heading (Θ) and curvature (K)
Output: A control set, E_x
$E_x = \emptyset$;
foreach $\theta_i, \theta_j \in \Theta$ *and* $\kappa_i, \kappa_j \in K$ **do**
  foreach x_f, y_f *s.t.* $L_\infty(O, [x_f, y_f]) = [1 \ldots \infty)$ **do**
    $u_i =$ trajectory$([0, 0, \theta_i, \kappa_i], [x_f, y_f, \theta_f, \kappa_f])$;
    if $u_i \neq \emptyset$ **then**
      $E_x \leftarrow u_i$;
      break;
    end
  end
end

Algorithm 1: A simple method of generating a control set.

B. Heuristic Cost Estimate

Heuristic estimates of the remaining cost in a partial plan are well-known to have the potential to focus the search enough to eliminate unnecessary computation while preserving the quality of the solution. The Euclidean distance metric is among the simplest options for a heuristic estimate of path length in the state lattice. This function is computationally efficient, and it satisfies the admissibility requirement of A* [18]. However, for differentially constrained planning, it is not a well-informed heuristic and, in the case of short paths, it can vastly underestimate the true path length, resulting in inefficient search. A heuristic for a vehicle with limited turning radius moving in the plane could be derived from the methods of Reeds and Shepp [19]. However, the Reeds-Shepp paths are discontinuous in curvature (i.e. infeasible to execute without stopping), and they do not account for discretization, so even these paths are underestimates. Given ample off-line computational resources, a straight-forward and effective way to predict path lengths is to pre-compute and store the actual cost heuristics that a planner will need, using the planner itself. Such a Heuristic Look-Up Table (HLUT) can be implemented as a database of real-valued query costs. Under this approach, the computation of the heuristic becomes a simple table dereference [20], [21].

IV. REACTIVE REPLANNING

The state lattice search space presented above is compatible with most dynamic programming algorithms. In order to achieve efficient implementation of efficient replanning algorithms (variants of D*), a number of implementation details are presented in this section.

A. Computing Edge Costs

The regularity of the state lattice allows an efficient optimization in evaluation of the cost of graph edges during planning with continuous cost maps, which is roughly equivalent in computational terms to pre-computing C space obstacles. Recall that, in continuous cost field environment models, the cost of a configuration is computed as a cost weighted swept volume (i.e. area in 2D workspace cost fields). That is, the sum of the workspace cell costs occupied by the vehicle volume. We denote the set of map cells occupied by the vehicle volume during execution of a particular motion as the *swath* of this motion. Since lattice edges repeat regularly, so do their associated swaths. Thus, it is possible to pre-compute the swaths for all elements of the control set. When costs change in the workspace cost map, the only computation required to update the cost of an edge (motion) is to add the costs of the cells in the swaths.

The top of Figure 3 depicts a motion of a tractor-trailer vehicle, along with the swath of this motion. In order to evaluate the cost of a motion, the costs of map cells in the swath (reproduced on the bottom of Figure 3) are simply summed up – an operation typically much more efficient than simulating the motion of the system. The simpler alternative of low-pass filtering the workspace cost map by a circular vehicle approximation will be significantly less accurate for systems with elongated shape. The calculation proceeds off-line for a state lattice and we care to satisfy differential constraints, so we use the correct vehicle shape and highly accurate simulation.

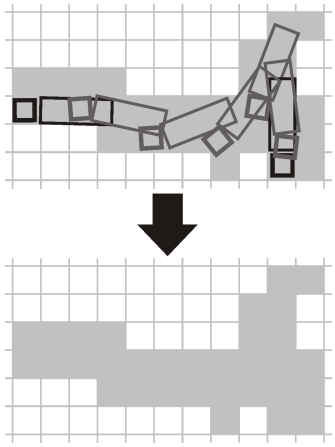


Fig. 3. An example of a pre-computed swath of a path for a tractor-trailer vehicle. Bottom: the swath allows computing the cost of a motion w.r.t. a cost map, without explicitly considering the motion itself (top).

B. Processing Edge Cost Updates in Replanning

D* variants were originally applied to grids [2], [3]. The earliest work on D* used the same resolution for both the cost map and the search space and implicit "edges" which connected states only to their nearest 8 neighbors. In this case, the mapping from a modified map cell to the affected search space edges and vertices is trivial. For a state lattice whose edges may span several map cells, the above historical simplifications of these issues are no longer feasible.

Suppose the replanner uses a priority queue to ensure optimality of the solution. For every change in the cost of the directed edge from the vertex x_i to x_j, $c(x_i, x_j)$, a replanning algorithm requires recomputing the cost of x_j and potentially inserting it into the priority queue. Assuming a map cell $m_{ij} \in \mathbb{N}^2$ changes cost, the planner needs to know the set of vertices V_c that potentially need to be re-inserted into the priority queue with new priority. Thus, the planner requires a mapping $Y : \mathbb{N}^2 \rightarrow V_c$.

To develop this mapping, we use the concept of swath, introduced in Section IV-A. More formally, we consider the

swath a set $C_s \subset \mathbb{N}^2$ of cost map cells that are occupied by the robot as it executes a motion. The cost of an edge that represents this motion is directly dependent on the costs of map cells in C_s. Recall that once we pre-compute the control set of a regular lattice, it is possible to pre-compute the swaths of the edges in it.

Since the mapping between edges and their terminal vertices is trivial, it is easier first to develop the mapping $Y' : \mathbb{N}^2 \rightarrow E_c$, where E_c is the set of edges that are affected by m_{ij} (i.e. the set of edges whose swaths pass through the cell). Determining Y' may still appear as a formidable task, given the high density of edges in the multi-dimensional state lattice. However, we again exploit the regularity of the lattice to simplify the problem. If we have $Y'' : O \rightarrow E_c$, where O is the map origin, then $Y' = Y'' + n, \forall n \in \mathbb{N}^2$. In other words, the set of edges, affected by $m_{ij} = O$ is identical for any other cell, up to the translation coordinates. Further, recall that the swath C_s of each edge in E_c is known. In principle, E_c contains all edges u_c, such that m_{ij} belongs to C_s of u_c. Hence, the mapping Y'' is exactly the set of edges, whose swaths pass through the 2D origin. The Figure 4 illustrates this idea. Like the control set and path swaths, the resulting set of edges can be pre-computed due to the regularity of the state lattice. An example of the V_c for the implementation described in Section VI is shown in the Figure 5.

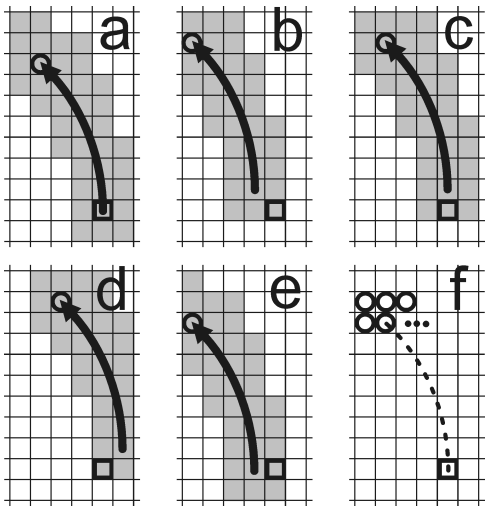


Fig. 4. The first several steps of pre-computing the list of graph vertices that are affected by a change in cost of a map cell. In a), a single element of a control set is chosen for this example. It emanates from the origin of the state lattice, thick square, and connects it to another graph vertex, thick circle. Grey cells are the swath of this motion. Suppose a map cell, located at the origin of the state lattice (thick square), changes cost. We attempt to find all translational versions of the chosen motion, whose swaths are affected by the changed map cell. In the subfigures b) – e), we iterate through several such translational versions of the motion. The resulting (edge end-point) vertices that are considered for insertion to the priority queue are shown in subfigure f). Typically, many more such vertices are processed for each edge (as suggested by ellipsis in subfigure f). The process repeats for all edges in the control set. Pre-computation allows eliminating any redundancy by generating a unique list of such vertices.

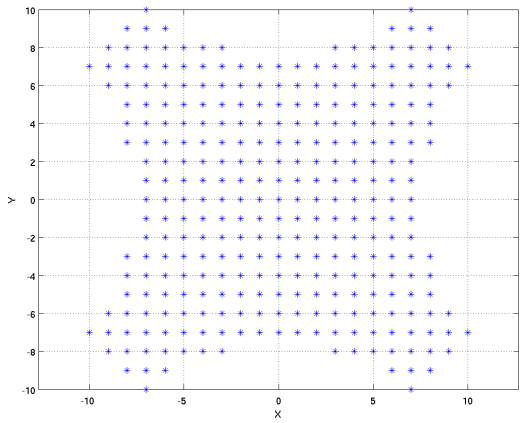


Fig. 5. A 2D projection of an example of V_c, the set of lattice states that are to be re-considered for every updated map cell. The units in the plot are state lattice cells. For the purposes of exposition, here the size of map cells is set to be equal to the size of state lattice (x, y) cells. For each map cell, m_{ij}, that changes cost, we place the set of vertices above in this Figure onto m_{ij} (i.e. the origin of the set of vertices, denoted with coordinates $(0, 0)$, is identified with m_{ij}). Next, we iterate through the depicted list of the vertices and place each one on the priority queue, if it was indeed affected by the cost change of m_{ij}.

V. FAST OPERATION

By virtue of the state lattice's general representation as a directed graph, it can be naturally extended with multi-resolution enhancements. Significant planning runtime improvement was achieved in the literature via a judicious use of the quality of representation of the planning problem, e.g. [22], [7], [8] among others. In field robotics, it is frequently beneficial to utilize a high fidelity of representation in the immediate vicinity of the robot (perhaps within its sensor range), and reduce the fidelity in the areas that are either less known or less relevant for the planning problem. Lower fidelity of representation is designed to increase search speed, while higher fidelity provides better quality solutions. Since grids have traditionally been utilized in replanning, the notion of varying the quality of problem representation has been identified with varying the resolution of the grid. However, our method varies the discretization of both the state and motions. We refer to managing the fidelity of state lattice representation as *graduated fidelity*.

In designing the connectivity of search space regions of different fidelities, care must be taken to ensure that all regions consist of motions that are feasible with respect to the robot's mobility model. If this rule is violated, mission failures become possible due to the differences in the representation of vehicle mobility. Figure 6 illustrates this situation using a simple example. Suppose a search space is used in which a high fidelity region of finite size surrounds and moves with the vehicle, and a disjoint lower fidelity grid is used beyond that. Suppose the A* algorithm is used to plan paths in this hybrid graph. A car-like robot attempts to travel to a goal on the other side of a collection of obstacles that forms a narrow corridor. As long as the low-fidelity region includes the corridor (black line), the planner will find a solution in the graph. However, the 90

degree turn in the path is actually infeasible, since the car-like robot cannot turn in place. As the vehicle moves, the high fidelity region will eventually include the turn in the corridor and the planner will then fail to find a solution. The only viable alternative will be to back up, thereby moving the corridor to the low fidelity region once again. Since the original state of the scenario has now been achieved, it is easy to see that this behavior will repeat forever. In order to avoid such difficulties, it is necessary to ascertain that all levels of fidelity include feasible motions. In particular, the connectivity of low fidelity regions must be a subset of that of the higher fidelity regions.

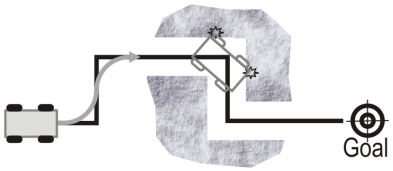


Fig. 6. A simple example of a motion planning problem, where a car-like robot that attempts to follow the infeasible path (black line) experiences a failure.

To implement graduated fidelity planning, the above design requires only a minor modification. Once the state lattice graph is separated into subgraphs of different fidelities as desired, each subgraph uses its own control set to achieve the chosen fidelity. Each control set defines the successors of a vertex being expanded during search. Care must be taken to design the control sets such that they adequately span the boundaries between the subgraphs. Note that control set design is the sole procedure needed to enable graduated fidelity. Replanning algorithms require no changes and will achieve the desired effects automatically.

It can be useful to enable a high fidelity subgraph to move along with the mobile robot as described in the example above. As shown in [23], such flexibility can be accomplished by undoing the effects of previous expansions of the vertices on the perimeter of the moving subgraph. Accomplishing this once again requires no change to the actual replanning algorithm. The change of graph connectivity that occurs between replans is presented to the planning algorithm as a change in cost of the affected graph vertices. Such topology based cost changes appear to replanning algorithms to be identical in nature to perception based cost changes. If the vertex expansion step is considered to be part of an external search space module, the planner actually cannot tell that the graph topology is changing.

More generally, it is straight-forward to extend the concept of graduated fidelity to allow multiple subgraphs of different fidelity to move or change shape between replans. Such flexibility results in a *dynamic search space*, which complements dynamic replanning algorithms to improve planning efficiency. Thus, the graduated fidelity extension of state lattice planning is conceptually simple and straight-forward to implement, and it can be designed to result in significant savings in runtime and memory usage in replanning.

VI. EXPERIMENTAL RESULTS

A differentially constrained motion planner, *lattice planner*, was implemented based on the state lattice and tested in a variety of scenarios, including in simulation and on real robots. The planner was ported to the VxWorks™hard real-time operating system that controls the JPL rover FIDO that was used in field experients. Figure 7 shows the results of a typical experiment with the FIDO running the lattice planner on-board to navigate autonomously amid dense rocks. In this experiment, the rover was given a command to drive to a goal 15 meters directly in front of it, as shown by the black line in the top of the Figure. This motion was infeasible due to large rock formations. However, the rover, under guidance of the lattice planner, negotiated this maze-like and previously unknown environment, and found a feasible path (white dots) to accomplish its mission, despite a very limited perception horizon of 3 meters and $\pm 40^\circ$ field of view.

We have not had a chance to optimize memory usage of our planner implementation; nevertheless, the peak memory usage of the lattice planner over all our experiments with the FIDO rover was less than 100MB. The bottom part of Figure 7 shows the semilog plot of the on-board lattice planner runtime per replan cycle, averaging at approximately 10Hz. This plot serves well to illustrate two points regarding typical planner runtime on-board FIDO: the computation time per replanning operation can vary greatly (depending on the difficulty of the problem at hand), and the replanning runtime was frequently lower than the time-resolution of the rover's operating system (5ms), which is observed via the bottom-limited segments of the plot.

Rover mobility was characterized by a minimum turning radius of 0.5m and a capacity of point turns, which had a high cost due to the time and energy required for reorienting wheels. Both cost map cells and (x, y)-cells of the state lattice were square with 20cm side length; both types of cells coincided in position. The rover used a single 1.6GHz CPU and 512MB of RAM, shared among all processes of the rover, including state estimation, stereo vision perception and communication systems.

The lattice planner utilized two fidelity regions. High fidelity region was square 21x21 mapcells (L_∞-radius of 2 meters), centered around the rover. It utilized a lattice control set with average outdegree 12. Its state space consisted of 2D position and heading (x, y, θ). It was generated using Algorithm 1. A trajectory generator in [17] was used to generate the motions between the given values of robot state. Motions were parameterized as cubic polynomial curvature, κ, functions of path length s, $\kappa(s)$. Low fidelity was represented as eight-connected grid.

In this experiment, the rover traversed approximately 30 meters and achieved the goal successfully (only the first two-thirds of the rover path are shown in the photograph due to the limited field of view of the external camera). No path tracking was used, and the rover executed verbatim the smooth and feasible motion computed by the lattice planner.

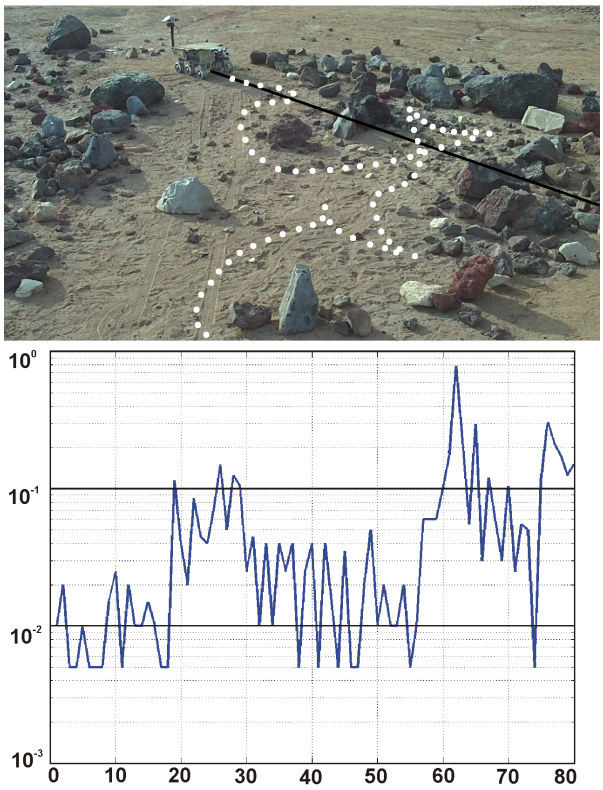


Fig. 7. A field experiment in the JPL Mars Yard. Top: the FIDO rover was commanded to go straight 15 meters (black line). The rover navigated autonomously among previously unknown maze-like obstacles, while running the graduated fidelity lattice planner on-board. White dotted line is the path traversed by FIDO. The rover encountered multiple difficult planning scenarios due to the very limited perception. It traveled approximately 30 meters in order to achieve its goal. Bottom: throughout numerous field experiments, lattice planner on-board FIDO averaged replanning frequency of approximately 10Hz.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper we described an effective approach to planning robot motions that satisfy differential constraints. In addition to leveraging dynamic replanning algorithms, this approach enables dynamic and deliberate changes in search space connectivity to boost efficiency. Standard replanning algorithms can be utilized, while the proposed search space design allows both the automatic satisfaction of differential constraints and the adjustment of the search space between replans. The method was successfully demonstrated in simulation and on real robots. Future work includes a further investigation into the state and motion space sampling to further improve planning efficiency.

REFERENCES

[1] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments: Part II," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, September 2008, pp. 1070–1076.

[2] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the Fourteenth International Joint Conf. on Artificial Intelligence*, August 1995.

[3] S. Koenig and M. Likhachev, "D* Lite," in *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, 2002.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[5] M. Pivtoraiko and A. Kelly, "Constrained motion planning in discrete state spaces," in *Field and Service Robotics*, vol. 25. Berlin / Heidelberg: Springer, July 2005, pp. 269–280.

[6] R. Bohlin, "Path planning in practice; lazy evaluation on a multi-resolution grid," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001.

[7] D. Ferguson and A. Stentz, "Multi-resolution Field D*," in *Proc. International Conference on Intelligent Autonomous Systems (IAS)*, 2006.

[8] D. Pai and L.-M. Reissell, "Multiresolution rough terrain motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 19–33, 1998.

[9] R. Szczerba, D. Chen, and J. Uhran, "Planning shortest paths among 2D and 3D weighted regions using framed-subspaces," *International Journal of Robotics Research*, vol. 17, no. 5, pp. 531–546, 1998.

[10] D. Hsu, R. Kindel, and J.-C. L. S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[11] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions*, pp. 293–308, 2001.

[12] J. Barraquand and J.-C. Latombe, "On nonholonomic mobile robots and optimal maneuvering," in *Proc. of the IEEE International Symposium on Intelligent Control*, 1989.

[13] S. Pancanti, L. Pallottino, and A. Bicchi, "Motion planning through symbols and lattices," in *Proc. of the Int. Conf. on Robotics and Automation*, 2004.

[14] 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.

[15] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proc. of the American Control Conference*, 2001.

[16] A. Bicchi, A. Marigo, and B. Piccoli, "On the reachability of quantized control systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 546–563, 2002.

[17] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *International Journal of Robotics Research*, vol. 22, no. 7/8, pp. 583–601, 2002.

[18] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Boston, MA: Addison-Wesley Longman Publishing Co., 1984.

[19] 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.

[20] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Optimal, smooth, nonholonomic mobile robot motion planning in state lattices," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-15, May 2007.

[21] R. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2006.

[22] R. Bohlin, "Path planning in practice; lazy evaluation on a multi-resolution grid," *Proc. of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2001.

[23] M. Pivtoraiko and A. Kelly, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.

Session IV

Motion planning

- **Title: ICS-AVOID, a Collision Avoidance Scheme for Dynamic Environments**
  Authors: Luis Martinez-Gomez and Thierry Fraichard

Benchmarking Collision Avoidance Schemes for Dynamic Environments

Luis Martinez-Gomez[†] and Thierry Fraichard[†]

***Abstract*— This paper evaluates and compare three state-of-the-art collision avoidance schemes designed to operate in dynamic environments. The first one is an extension of the popular Dynamic Window approach; it is henceforth called TVDW which stands for *Time-Varying Dynamic Window*. The second one called NLVO builds upon the concept of *Non Linear Velocity Obstacle* which is a generalization of the Velocity Obstacle concept. The last one is called ICS-AVOID, it draws upon the concept of *Inevitable Collision States*, *ie* states for which, no matter what the future trajectory of the robotic system is, a collision eventually occurs. The results obtained show that, when provided with the same amount of information about the future evolution of the environment, ICS-AVOID outperforms the other two schemes. The primary reason for this has to do with the extent to which each collision avoidance scheme reasons about the future. The second reason has to do with the ability of each collision avoidance scheme to find a safe control if one exists. ICS-AVOID is the only one which is complete in this respect thanks to the concept of Safe Control Kernel.**

***Index Terms*— Motion Safety; Collision Avoidance; Dynamic Environments; Inevitable Collision States, Velocity Obstacles, Dynamic Window.**

I. INTRODUCTION

A. Background and Motivations

Autonomous mobile robots/vehicles navigation has a long history by now. Remember Shakey's pioneering efforts in the late sixties [1]. Today, the situation has dramatically changed as illustrated rather brilliantly by the 2007 DARPA Urban Challenge[1]. The challenge called for autonomous car-like vehicles to drive 96 kilometers through an urban environment amidst other vehicles (11 self-driving and 50 human-driven). Six autonomous vehicles finished the race thus proving that autonomous urban driving could become a reality. Note however that, despite their strengths, the Urban Challenge vehicles have not yet met the challenge of fully autonomous urban driving (how about handling traffic lights or pedestrians for instance?).

Another point worth mentioning is that at least one collision took place between two competitors. This unfortunate mishap raises the important issue of *motion safety*, *ie* the ability for an autonomous robotic system to avoid collision with the objects of its environment. The size and the dynamics of the Urban Challenge vehicles make them potentially dangerous for themselves and their environment (especially when driving at high-speed). Therefore, before letting such autonomous systems transport around or move among people, it is vital to assert their ability to avoid collisions.

In the last forty years, the number and variety of autonomous navigation schemes that have been proposed is huge (*cf* [2]). In general, these navigation schemes intend to fulfill two key purposes: reach a goal and avoid collision with the objects of the environment. When it comes to collision avoidance, once again, many collision avoidance schemes have been proposed. Their aim of course is to ensure the robotic systems' safety. However, the analysis carried out in [3] of the most prominent navigation schemes (*ie* the ones currently used by robotics systems operating in real environments, *eg* [4]–[7]) shows that, especially in environments featuring moving objects, *motion safety is not guaranteed* (in the sense that collisions can occur even if they have full knowledge of the environment future evolution: no uncertainty or spurious information). As shown in [3], collision avoidance in dynamic environments is complex since it requires to explicitly reason about the *future behaviour* of the moving objects with a *time horizon*, *ie* the duration over which the future is taken into account, which is determined by the nature of both the moving objects and the robotic system at hand. Failure to do so yields collision avoidance schemes with insufficient motion safety guarantees.

B. Contributions

The primary purpose of this paper is to explore this time horizon issue and to show how important it is in the design of a truly safe collision avoidance scheme. To that end, this paper will evaluate and compare three state-of-the-art collision avoidance schemes that have been explicitly designed to handle dynamic environments. The first one is from [8] and is henceforth called *Time-Varying Dynamic Window* (TVDW), it is a straightforward extension of the popular Dynamic Window approach [6]. The second one builds upon the concept of *Non Linear Velocity Obstacle* (NLVO) [9] which is a generalization of the Velocity Obstacle concept [7]. The last one, ICS-AVOID [10], draws upon the concept of *Inevitable Collision States* developed in [11] (*aka* Obstacle Shadow [12] or Region of Inevitable Collision [13], [14]). The three collision avoidance schemes do reason about the future evolution of the environments but they do so differently, each scheme has its own time horizon.

When placed in the same environment and provided with exactly the same amount of information about the future, the results we have obtained show that ICS-AVOID performs significantly better than the other two schemes.

[†]INRIA, CNRS-LIG & Grenoble University, France.

[1]`http://www.darpa.mil/grandchallenge`.

The primary reason for this has to do with the way each collision avoidance scheme uses the information about the future, thus emphasizing the fact that, reasoning about the future is not nearly enough, it must be done with an appropriate time horizon. In contrast with TVDW and NLVO, ICS-AVOID is the only scheme that reasons over an infinite time-horizon. The analysis carried out in [10] shows that if ICS-AVOID were provided with full knowledge about the future, it would guarantee motion safety no matter what. Now, it could be argued that infinite knowledge about the future is not available in realistic cases (which is true). The fact remains that ICS-AVOID is the only scheme that is able to make full use of all the information about the future which is provided.

The second reason has to do with the decision part of each collision avoidance scheme. In all cases, their operating principle is to first characterize forbidden regions in a given control space and then select an admissible control, *ie* one which is not forbidden. Accordingly motion safety also depends on the ability of the collision avoidance scheme at hand to find such admissible control. In the absence of a formal characterization of the forbidden regions, all schemes resort to sampling (with the inherent risk of missing the admissible regions). In contrast, ICS-AVOID through the concept of *Safe Control Kernel* is the only one for which it is guaranteed that, if an admissible control exists, it will be part of the sampling set.

C. Outline of the Paper

The paper is organized as follows: Section II gives an overview of the collision avoidance schemes used for the comparative evaluation: TVDW, NLVO and ICS-AVOID. Afterwards, Section III details the way each collision avoidance scheme reasons about the future. Section IV describes the benchmarking and simulation setup. The benchmark results are presented in Section V. Discussion and concluding remarks are made in Section VI.

II. STATE-OF-THE-ART COLLISION AVOIDANCE SCHEMES

As exposed in the introduction, the benchmarking concerns TVDW, NLVO and ICS-AVOID. The first two are extensions to popular collision avoidance schemes used in real-world applications: Dynamic Window (DW) and Velocity Obstacles (VO). DW has been demonstrated at relatively high speeds (up to $1\ m/s$) in complex environments with Minerva [15], Rhino [16] and Robox [17], robotic tour-guides that have operated for different time periods in different places in the United States, Germany and Switzerland. VO has been tested with MAid [18], an automated wheelchair that navigated in the concourse of the central station in Ulm (DE) and during the German exhibition Hanover Fair'98. ICS-AVOID, is the continuation of the work done around the ICS concept for safe motion planning in dynamic environments [19], [20] with applications in driverless vehicles [21], [22].

A. Time Varying Dynamic Window

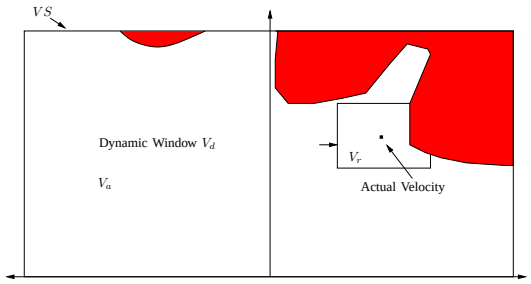
The Dynamic Window approach is a velocity space based local reactive avoidance scheme where search for admissible controls is carried out directly in the linear and angular velocity space [6]. The search space is reduced by the system kinematic and dynamic constraints to a set of reachable velocities (V_r) in a short time interval (Δt) around the current velocity vector (Fig.1a):

$$V_r = \{(v,\omega) | v \in [v_c - \dot{v_b}\Delta t, v_c + \dot{v_a}\Delta t] \wedge \omega \in [\omega_c - \dot{\omega_b}\Delta t, \omega_c + \dot{\omega_a}\Delta t]\} \quad (1)$$

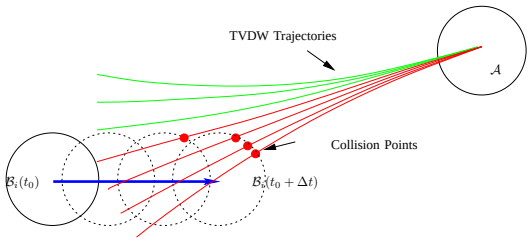
where $\dot{v_a}$, $\dot{\omega_a}$, $\dot{v_b}$ and $\dot{\omega_b}$ are maximal translational/rotational accelerations and breaking decelerations. A velocity is admissible (V_a) if it allows the system to stop before hitting an object:

$$V_a = \{v, \omega \leq \sqrt{2\rho_{min}(v,\omega)\dot{v_b}} \wedge \sqrt{2\rho_{min}(v,\omega)\dot{\omega_b}}\} \quad (2)$$

An admissible velocity optimizing a given cost function is selected at each time step. This approach considers the objects in the environment as static. TVDW extends this scheme by calculating at each instant a set of immediate future obstacles trajectories in order to check for collision in the short term [8]. In this respect TVDW is superior to DW because it reasons about the future behaviour of the obstacles. The extent of the look ahead time is set to equal the time it takes to the robotic system to stop, if no collision occurs during that time the velocity is considered admissible (Fig.1b).



(a) Dynamic Window.



(b) Time-Varying Dynamic Window.

Fig. 1: Dynamic Window based approaches.

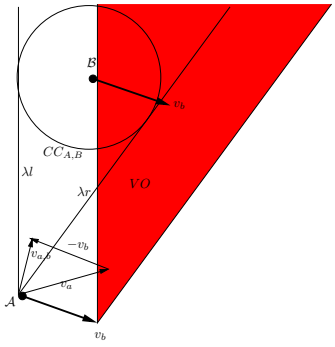
B. Non-Linear Velocity Obstacles

Velocity Obstacles is a reactive approach that operates in the Cartesian velocity space of the robotic system considered [7]. VO takes into account the velocity of the moving objects (assumed to be moving with a constant linear

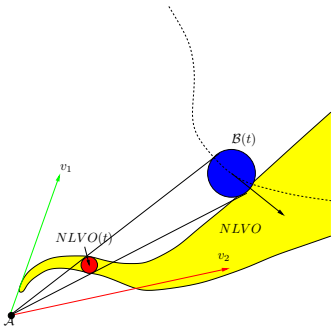
velocity). Each object yields a set of forbidden velocities whose shape is that of a cone (Fig.2a depicts the linear velocity space of the robotic system, the red conical region on the right is the set of forbidden velocities that would yield a collision between the robot $\mathcal{A}$ and the moving object $\mathcal{B}$). Should the robotic system select a forbidden velocity, it would collide with the moving object at a later time (possibly infinite) in the future. In practice, velocities yielding a collision occurring after a given time horizon (t_H) are considered as admissible. NLVO is an extension of VO that considers known arbitrary velocity profiles for the moving objects [9]. NLVO consist of all velocities of $\mathcal{A}$ at t_0 that would result in collision with $\mathcal{B}$ at any time $t_0 \leq t \leq t_H$. As depicted in Figure.2b, $NLVO(t)$ is a scaled down $\mathcal{B}$, bounded by the cone formed between $\mathcal{A}$ and $\mathcal{B}(t)$, thus, NLVO is a warped cone with apex at $\mathcal{A}$ and formally defined as:

$$NLVO = \bigcup_{t_0 \leq t \leq t_H} \frac{\mathcal{B}(t)}{t - t_0} \tag{3}$$

where $\frac{\mathcal{B}(t)}{t-t_0}$ is the set $\mathcal{B}(t)$ scale down by $(t - t_0)$. One issue (often overlooked) with the VO representation is that, in a closed environment, every velocity is forbidden since it eventually yield a collision. For that reason, both VO and NLVO require a time horizon t_H that cannot be arbitrarily large.



(a) Velocity Obstacles.



(b) Non Linear Velocity Obstacles.

Fig. 2: Velocity Obstacles based approaches.

C. ICS-AVOID

ICS-AVOID is a reactive navigation approach based upon the concept of Inevitable Collision State (ICS) [11]. An ICS is defined as a state for which, no matter what the future trajectory followed by the system is, a collision eventually occurs. ICS-AVOID searches the control space of the system for a control to apply at the next time step. A control is admissible if it drives the system to a non-ICS state. To test for ICS-ness the ICS-Checker presented in [23] is used. If the current state is not an ICS then it is guaranteed that ICS-AVOID will find and return an admissible control (Safe Control Kernel) [10].

III. REASONING ABOUT THE FUTURE

All the collision avoidance schemes used in the benchmarking make use of a model of the future, that is, they take into account the future behaviour of the obstacles in the environment. The different extent in which they use the available information have an impact in the decisions they made and consequently in their overall performance. TVDW considers as look ahead the braking time t_B (the time it takes to the system to go from its current velocity to a halt). This time is then state dependent and upper-bounded. NLVO use as look ahead an arbitrarily set time horizon (t_H), in other words, there is no clear guideline on how to choose it an is not a function of the system dynamics nor current state. Furthermore, it can't be set to a very large value because in closed environments it will render all velocities inadmissibles. ICS-AVOID in accordance to the ICS defintion reasons in terms of infinite duration. It uses the available information about the unfolding of the environment up to infinity. The different look ahead of the collision avoidance schemes is illustrated in Fig.3 to emphasize the fact that both TVDW and NLVO truncate their future model and disregard any information beyond t_B and t_H respectively (even if it's available). In contrast, this isn't the case for ICS-AVOID.

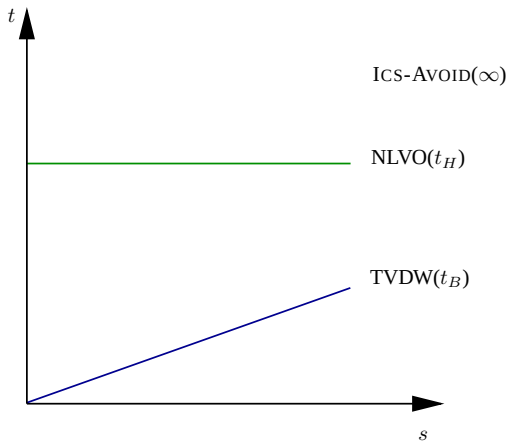


Fig. 3: Look-ahead of the different schemes.

IV. BENCHMARK AND SIMULATION SETUP

To assess the performance of the collision avoidance schemes just presented a comparative evaluation was conducted. A simulation environment capable of reproducing the same conditions for all the schemes was chosen to conduce

the benchmarking. The robotic system, environment setup and implementation is discussed next.

1) Robotic System: Point Mass Model: Let $\mathcal{A}$ be modeled as a disk with point mass non-dissipative dynamics. A *state* of $\mathcal{A}$ is defined as $s = (x, y, v_x, v_y)$ where (x, y) are the coordinates of the center of the disk and v_x, v_y are the axial components of the velocity. A control of $\mathcal{A}$ is defined by the pair (u_x, u_y) which denote the force exerted by the actuators along the x- and y-axis respectively. The motion of $\mathcal{A}$ is governed by the following differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v_x} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_y \qquad (4)$$

with a bound in the control given by the maximum acceleration: $\frac{u_x^2+u_y^2}{m^2} \leq a_{max}^2$ where m is the robot mass.

2) Workspace Model: $\mathcal{A}$ moves in a closed 2D workspace $\mathcal{W}$ (100 by 100 meters), cluttered up with disk-shaped moving objects (grown by the radius of $\mathcal{A}$). A total of twenty three objects move with random constant speeds (between 1 to 10 m/s) along complex cyclic trajectories (closed B-splines with 10 random control knots). Figure 4 shows the trajectories of the objects to illustrate the complexity of the environment. This setup can theoretically provide future

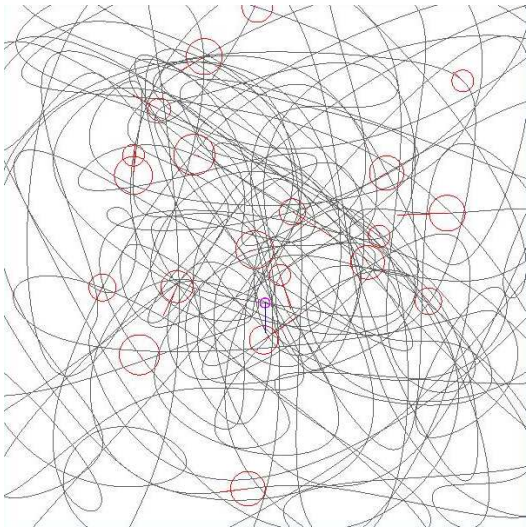


Fig. 4: Workspace example, 23 obstacles (represented by circles) with random generated velocities and B-Splines trajectories.

information about the behaviour of the moving objects up to infinity. In practice, knowledge is provided until a fixed time in the future t_F after which constant linear motion is assumed (Fig. 5). This to resemble realistic cases where prediction quality degrades as time pass by.

3) Implementation: The simulation environment and collision schemes were programmed entirely in C++ using OpenGL as rendering engine. The random number generator employed to produce the obstacles trajectories and velocities was seeded with a set of identical numbers to achieve

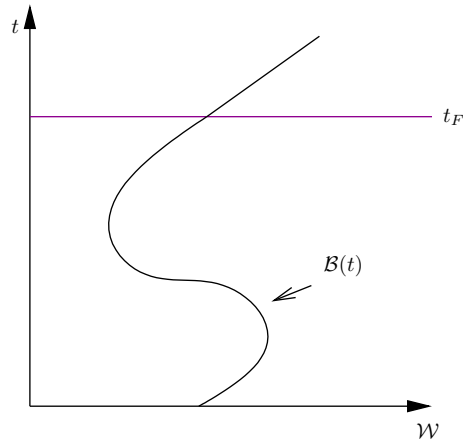


Fig. 5: World Model of the future.

an identical reproduction of simulation conditions for each of the collision avoidance schemes in the benchmark. The information about the future behaviour of the objects in the environment was made available to all the schemes with a limit of $t_F = 1$, 3 and 5 seconds into the future.

V. Benchmark

The collision avoidance schemes were tested on a set of five runs with a duration of two minutes each. We varied the amount of available information about the future behaviour of the obstacles in the environment with $t_F = 1$, 3 and 5 seconds. For each run the number of collisions between $\mathcal{A}$ and the objects $\mathcal{B}_i$ are recorded in Table I.

Scheme	Run	Collisions TF=1(s)	Collisions TF=3(s)	Collisions TF=5(s)
TVDW	1	5	6	3
	2	12	4	4
	3	5	7	3
	4	12	2	4
	5	12	2	4
Average:		9.2	4.2	3.6
NLVO	1	10	2	0
	2	8	2	0
	3	12	2	0
	4	3	3	2
	5	7	2	2
Average:		8.0	2.2	0.8
ICS-AVOID	1	7	0	0
	2	0	0	0
	3	1	0	0
	4	1	0	0
	5	1	0	0
Average:		2.0	0.0	0.0

TABLE I: Benchmarking of collision avoidance schemes.

TVDW (Fig. 6) performs poorly in comparison with the other two schemes. One of the main causes of failure is the limited extent in which the scheme use the information available about the future trajectories of the objects: as explained before it limits itself to a small fraction of the time at hand (t_B). In contrast, NLVO (Fig. 7) exploits better the given information. In these runs t_H was set equal to t_F so all

the available information could be taken into account. NLVO averages less of one collision per run in the 5 second setup, nonetheless, it fails to guarantee the safety of the system when provided with less information. ICS-AVOID (Fig. 8) has the best performance in all the time setups. ICS-AVOID is designed to reason in terms of infinite duration but even when dealing with minimal information about the future (1 second) it outperfomed the other two schemes. When given more information (3 and 5 seconds) not a single collision occured. The results show the importance of the look ahead time, when a colllision avoidance scheme disregard available information its performance is lower compared to those that use more.

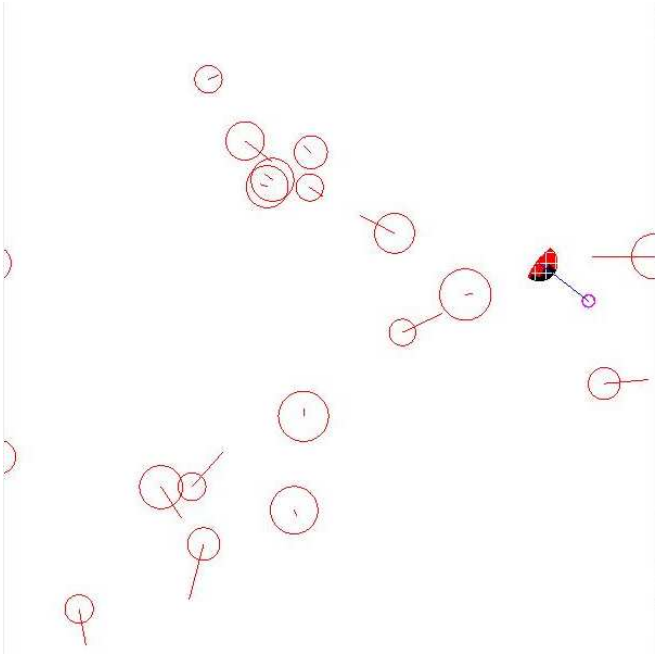


Fig. 6: TVDW. Admissible velocities (V_a) are represented in black, velocities in red are forbidden.

VI. CONCLUSION

We have presented a comparative evaluation with three state-of-the-art collision avoidance schemes designed to handle complex dynamic environments. The results show that, when provided with the same amount of information about the future evolution of the environment, ICS-AVOID outperforms the others. The reason for this has to do with the extent to which each collision avoidance scheme reasons about the future.

ACKNOWLEDGEMENTS

This work has been partially supported by the European Commission contract "Have-It FP7-IST-2007-212154".

REFERENCES

[1] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Menlo Park, CA (US), Technical note 323, Apr. 1984.

[2] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

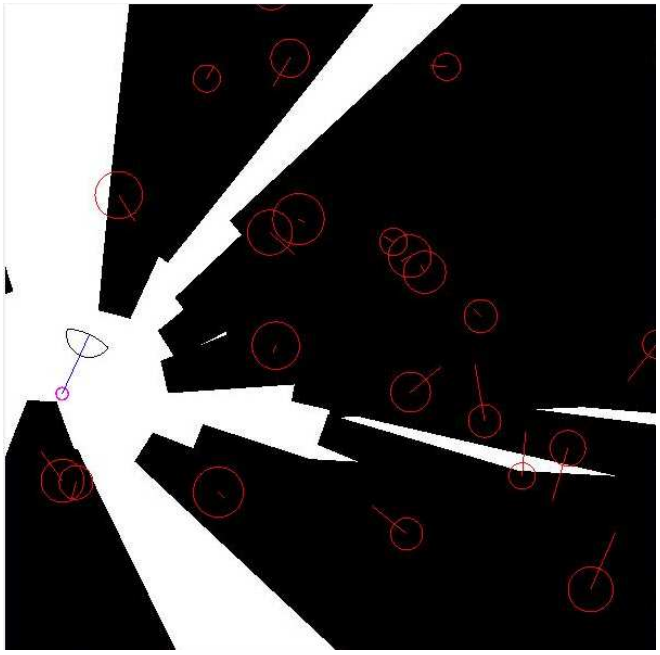


Fig. 7: NLVO. Black warped cones are forbidden velocities for the robotic system.

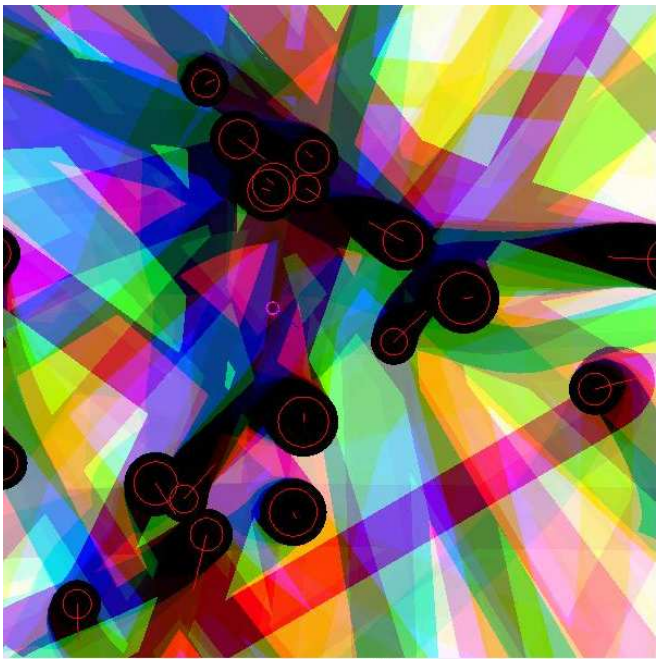


Fig. 8: ICS-AVOID. Black regions are forbidden states (ICS).

[3] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma (IT), Apr. 2007.

[4] J. Borenstein and Y. Korem, "The vector field histogram — fast obstacle avoidance for mobile robts," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, June 1991.

[5] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, Feb. 2004.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, Mar. 1997.

[7] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, July 1998.

[8] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," *Proc. of the Int. Conf. on Robotics and Automation*, Apr. 2007.

[9] F. Large, C. Laugier, and Z. Shiller, "Navigation among moving obstacles using the NLVO : Principles and applications to intelligent vehicles," *Autonomous Robots Journal*, vol. 19, no. 2, pp. 159–171, September 2005.

[10] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Kobe (JP), May 2009.

[11] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[12] J. Reif and M. sharir, "Motion planning in the presence of moving obstacles," in *Proc. of the IEEE Int. Symp. on Foundations of Computer Science*, Portland, OR (US), Oct. 1985.

[13] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[14] N. Chan, M. Zucker, and J. Kuffner, "Towards safe motion planning for dynamic systems using regions of inevitable collision," in *Proc. of the workshop on Collision-free Motion Planning for Dynamic Systems*, Rome (IT), Apr. 2007, workshop held in association with the IEEE Int. Conf. on Robotics and Automation.

[15] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second generation mobile tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[16] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1-2, 2000.

[17] R. Philippsen and R. Siegwart, "Smooth and efficient obstacle avoidance for a tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Taiwan (TW), Sept. 2003.

[18] E. Prassler, J. Scholz, and P. Fiorini, "A robotic wheelchair for crowded public environments," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 38–45, Mar. 2001.

[19] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB (CA), Aug. 2005.

[20] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma (IT), Apr. 2007.

[21] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Toward urban driverless vehicles," *Int. Journal of Vehicle Autonomous Systems*, vol. 6, no. 1/2, 2008.

[22] L. Bouraoui, S. Petti, A. Laouiti, T. Fraichard, and M. Parent, "Cybercar cooperation for safe intersections," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, Toronto, ON (CA), September 2006.

[23] L. Martinez-Gomez and T. Fraichard, "An efficient and generic 2d inevitable collision state-checker," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Nice (FR), Sept. 2008.

Session IV

Motion planning

- **Title: Mapping Obstacles to Collision States for On-line Motion Planning in Dynamic Environments**
  Authors: Oren Gal and Zvi Shiller

Mapping Obstacles to Collision States for On-line Motion Planning in Dynamic Environments

Oren Gal and Zvi Shiller

***Abstract*— This paper presents a representation of static and moving obstacles, using Velocity Obstacles (VO), for on-line planning in dynamic environments. Each obstacle is mapped to forbidden states by selecting a proper time horizon for the velocity obstacle. The proper choice of the time horizon ensures that the boundary of the mapped obstacle overlaps with the boundary of the set of inevitable collision states (ICS). This time horizon is determined by the minimum time it would take the robot to avoid collision, either by stopping or by passing the respective obstacle. This representation allows safe on-line planning using only one step look ahead. The on-line trajectories favorably compare with the trajectories obtained by a global planner.**

I. Introduction

Most of the work on motion planning over the past twenty years has focused on static obstacles, with a few exceptions. We distinguish between local and global planners. The local planner generates one, or a few steps at every time step, whereas the global planner uses a global search to the goal over a time spanned tree. Examples of local (reactive) planners are [3], [16], [8], [11], but most do not guarantee safety as they are too slow and hence their ability to look-ahead and avoid states of inevitable collision is very limited. Recently, iterative planners [5], [7], [1], [12], [10], [15] were developed that compute several steps at a time, subject to the available computation time. The trajectory is generated incrementally by exploring a search-tree and choosing the best branch. These planners too do not address the issue of safety.

Only a few works have addressed the safety issue in dynamic environments, which is crucial for partial (local) planning. One approach is to use braking policies [17]; another is to ensure local avoidance for a limited time [10]. However, neither considers the dynamic of the moving robot. A promising approach to safe motion planning in dynamic environment is the consideration of "regions of inevitable collision," first introduced in [9] and later extended in [6], [14], [4], [2].

We address the issue of safety for an on-line local planner in dynamic environments. Motion safety is guaranteed by ensuring that the robot's velocity does not penetrate the velocity obstacle, which is generated for a carefully selected time horizon. This velocity obstacle is a mapping of the obstacle to a set of forbidden states, which overlaps with the boundary of the set of inevitable collision states. Repelling the robot's velocity from entering the inevitable collision states ensures (if a solution exists) that the robot does not crash into any static or moving obstacle. The safe time horizon, which is obstacle specific, is determined by computing the minimum time it would take the robot to avoid collision, either by stopping or by passing the respective obstacle. Determining the safe time horizon is computationally very efficient and it does not require a prior mapping of inevitable collision states. We demonstrate the approach for on-line motion planning in very crowded static and dynamic environments.

This work was performed at the Paslin Robotics Research Laboratory at the Ariel University Center, Israel.

Oren Gal is a graduate student at the Department of Mechanical Engineering, Technion, Israel. (orengal@tx.technion.ac.il)

Zvi Shiller is with the Department of Mechanical Engineering, Ariel University Center, Israel (shiller@ariel.ac.il)

II. The Velocity Obstacle

The velocity obstacle represents the set of all colliding velocities of the robot with each of the neighboring obstacles. It maps static and moving obstacles into the robot's velocity space. The velocity obstacle (VO) of a planar circular obstacle, B, that is moving at a constant velocity v_b, is a cone in the velocity space of robot A, reduced to a point by enlarging respectively obstacle B. Each point in VO represents a velocity vector that originates at A. Any velocity of A that penetrates VO is a colliding velocity that would result in collision between A and B at some future time. All velocities of A that are outside of VO are safe as long as B stays on its current course. The velocity obstacle thus allows determining if a given velocity is potentially dangerous, and suggesting possible changes to this velocity to avert collision. If B is known to move along a curved trajectory or at varying speeds, it would be best represented by the nonlinear velocity obstacle (NLVO), which accounts for a general trajectory of the obstacle, while assuming a constant velocity of the robot. It applies to the situation where, at time t_0, a point A attempts to avoid an obstacle, B, that is following a general known trajectory, $c(t)$, and at time t_0 is located at $c(t_0)$. The $NLVO$ consists of all velocities of A at t_0 that would result in collision with the obstacle at any time $t > t_0$. Selecting a single velocity, v_a, at time $t = t_0$ outside the $NLVO$ thus guarantees that A avoids collision at all times. It is constructed as a union of its temporal elements, $NLVO(t)$, which is the set of all absolute velocities of A, v_a, that would result in collision at a specific time t.

The velocity v_a that would result in collision with point p in B at time $t > t_0$, expressed in a frame centered at $A(t_0)$, is simply

$$v_a = \frac{c(t) + r}{t - t_0}, \tag{1}$$

where r is the vector to point p in the obstacle's fixed frame. The set, $NLVO(t)$ of all absolute velocities of A that would

result in collision with any point in B at time $t > t_0$ is obtained by replacing r with the set of all points in B:

$$NLVO(t) = \frac{c(t) \oplus \mathscr{B}}{t - t_0}. \tag{2}$$

where $\mathscr{B}$ represents the set of all points in the grown obstacle B, defined relative to some center that follows the curve $c(t)$, and $\oplus$ represents the Minkowski sum. Clearly, $NLVO(t)$ is a scaled $\mathscr{B}$, located at a distance from A that is inversely proportional to the collision time t. The entire $NLVO$ is the union of its temporal subsets from t_0, the current time, to some set time horizon t_h:

$$NLVO_{t_0}^{t_h} = \bigcup_{t_h > t > t_0} \frac{c(t) \oplus \mathscr{B}}{t - t_0}. \tag{3}$$

The non-linear v-obstacle is a warped cone. If $c(t)$ is bounded over $t = (t_0, \infty)$, then the apex of this cone is at $A(t_0)$. The boundaries of the $NLVO$ represent velocities that would result in A grazing B. The smallest safe time horizon is the one that allows sufficient time to avoid or escape collision as discussed next.

III. Time Horizon

The time horizon plays an important role in selecting feasible avoidance maneuvers. It allows considering only those maneuvers that would result in a collision within a specified time interval. Setting the time horizon too high would be too prohibitive, as it would mark as dangerous maneuvers resulting in collision at a distant time; selecting a too small time horizon would permit dangerous maneuvers that are too close and at too high speeds to avoid the obstacle. It is essential that the proper time horizon be selected to ensure that a safe maneuver, even if temporarily pointing toward the obstacle, is selected. The smallest safe time horizon is the one that allows sufficient time for the robot to avoid the obstacle either by stopping or by passing. It depends on the size of the obstacle, its velocity, and the robot's dynamic constraints.

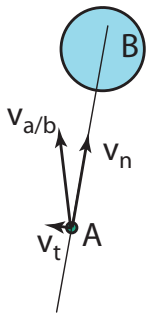


Fig. 1. The robot and obstacle on a collision course

Consider a robot A and an obstacle B, each moving at some constant velocity. The time horizon is relevant only if the two are on a collision course, i.e. the velocity v_a penetrates the velocity obstacle of B. To determine the proper time horizon for this case, we first transform the problem into the avoidance of a static obstacle by considering the relative velocity $v_{a/b}$, as shown in Figure 1. The relative velocity $v_{a/b}$ is then projected into two components, v_n and v_t that are parallel and normal to the line connecting A and the center of B, as shown in Figure 1.

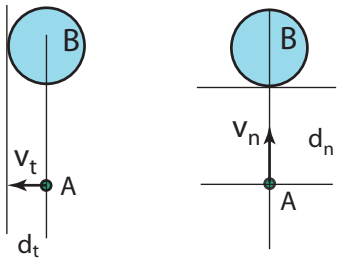


Fig. 2. Stopping and passing maneuvers

The robot can avoid collision by either stopping before hitting the obstacle, or by passing it on either side. To stop, the robot's longitudinal velocity v_n must decelerate to zero before traversing the distance d_n; to pass, the robot must traverse the lateral distance d_t faster than it would traverse the longitudinal distance d_n. We select the time horizon such that when the robot's velocity first penetrates the velocity obstacle, it still has sufficient time to avoid collision either by stopping *or* by passing. To this end, we wish to determine the minimum time required for each maneuver (stopping and passing) to select the smallest safe time horizon. For simplicity, we decouple the two maneuvers, assuming that each is executed by a single control effort. The minimum times for the stopping and passing maneuvers thus depend on the initial velocity, distance, and the control constraint in each direction. The smallest safe time horizon is then the smallest of the minimum times for stopping and passing.

A. Stopping time

The minimum stopping time is the time it would take the robot to decelerate to a stop from its current normal velocity v_n, using the maximum deceleration. Assuming a constant longitudinal deceleration, $u_n < 0$, the stopping time is

$$t_{stop} = \frac{v_n}{-u_n}. \tag{4}$$

Since the VO assumes collision at a constant speed, whereas t_{stop} assumes a constant deceleration, using t_{stop} as the time horizon would alert the robot too early of a potential collision. Taking into account the decelerating velocity allows us to use a shorter time horizon. To determine how short, we compare the distance traveled over t_{stop} at a constant speed and at a constant deceleration.

The distance traveled at a constant velocity v_n over the stopping time t_{stop} is:

$$d_{const} = v_n t_{stop}. \tag{5}$$

The distance traveled at a constant deceleration u_n from the initial velocity v_n to a stop is:

$$d_{dec} = v_n t_{stop} + \frac{1}{2} u_n t_{stop}^2. \tag{6}$$

Substituting $v_n = -u_n t_{stop}$ into (6) yields:

$$d_{dec} = v_n t_{stop} - \frac{1}{2} v_n t_{stop} = \frac{1}{2} v_n t_{stop} = \frac{1}{2} d_{const}. \quad (7)$$

Since the distance traveled at a constant deceleration is half the distance traveled at a constant v_n over the stopping time t_{stop}, the moving robot should start decelerating when the time to collision at a constant speed drops to half the stopping time (4). The smallest time horizon t_s for the stopping maneuver is therefore half the stopping time t_{stop}:

$$t_s = \frac{1}{2} t_{stop} = \frac{v_n}{-2u_n}. \quad (8)$$

B. Passing time

The minimum time for passing, t_p, is the solution to the minimum time problem of traversing the distance d_t, given an initial velocity v_t and an unspecified final velocity. The solution to this problem is an extremal control that either accelerates or decelerates, depending on the signs of d_t and v_t.

The velocity v_f developed by accelerating at u_t over t_p until traversing d_t satisfies:

$$v_f = v_t + u_t t_p \quad (9)$$

$$v_f^2 = v_t^2 + 2u_t d_t. \quad (10)$$

The minimum time, t_p, to traverse the distance d_t is thus the smallest positive solution:

$$t_p = min \frac{-v_t \pm \sqrt{v_t^2 + 2u_t d_t}}{u_t} \quad (11)$$

Note that there are two such solutions, one for passing on the right and one on the left. Obviously, the smallest of the two is selected.

Selecting the time horizon as the smallest of the two times

$$t_h = min\{t_s, t_p\} \quad (12)$$

ensures that when the robot's velocity touches the boundary of the velocity obstacle, there remains sufficient time to avoid the obstacle either by stopping *or* by passing. Penetrating the velocity obstacle would leave no time for a safe avoidance maneuver, which implies that the boundary of the velocity obstacle, generated for $t \leq t_h$, represents states on the boundary of the *ICS*. The time horizon is computed individually for each obstacle, using the relative velocity between the robot and obstacle.

C. A compact representation of velocity obstacles

There is no need to compute the entire velocity obstacle for $t \in (t_0, t_h)$ since generally, any collision at time $t < t_h$ would be the result of v_a first penetrating the temporal velocity obstacle at the time horizon, $NLVO(t_h)$:

$$NLVO(t_h) = \frac{c(t_h) \oplus \mathscr{B}}{t_h - t_0}. \quad (13)$$

Thus, each obstacle is mapped to one temporal *VO*, $NLVO(t_h)$, which is a set of a similar shape to the robot but of a different size and location. This greatly reduces computation time and provides an intuitive mapping of the dynamic environment. Thus, the original collision avoidance problem turns into the velocity avoidance of the mapped obstacles.

IV. The Planner

The efficient representation of static and moving obstacles by velocity obstacles allows us to efficiently plan safe trajectories in dynamic environments. The proper choice of the time horizon ensures survival of the robot, i.e. not entering inevitable collision states (ICS). For one obstacle, this guarantees convergence to the goal. For many obstacles, a solution cannot be guaranteed due to the changing nature of the environment. The computational effort is drastically reduced by considering only "safe" attainable states that satisfy system dynamics and are out of the ICS.

A. System Dynamics

For simplicity, the robot is assumed a planar point mass. This is necessary for computational reasons, and is in no way a limitation of this approach.

We consider the following point mass model:

$$\ddot{x} = u_1; |u_1| \leq 1 \quad (14)$$

$$\ddot{y} = u_2; |u_2| \leq 1 \quad (15)$$

where $(x,y)^T \in R^2$ represents the robot's position in a Cartesian coordinate frame and $(u_1, u_2)^T \in R^2$ represents the robot's controls.

B. Attainable Cartesian Velocities

Given the robot's dynamics, we wish to compute the set of attainable Cartesian velocities (*ACV*) of the maneuvering robot that can be reached over a given time interval, Δt [13]. This set contains the avoidance maneuvers that are dynamically feasible from a given state. The attainable *Cartesian* velocities, $ACV(t + \Delta t)$ are integrated from the current state $(x,v) = (x,y,\dot{x},\dot{y})$ by applying all admissible controls $u = (u_1, u_2) \in U$:

$$ACV(t + \Delta t) = \{v | v = v(t) + \Delta t u, u \in U\}. \quad (16)$$

The geometric shape of $ACV(t + \Delta t)$ depends on the specific system dynamics. For a point mass model, with constant control constraints, it is a rectangle, similar in shape to the set of admissible controls U, as shown in Figure 3.

C. Tree Search

The planner uses a depth first A* search over a tree that expands over time to the goal. Each node contains the current position and velocity of the robot at the current time step. At each state, the planner computes the set of admissible velocities *ACV*, which is then tessellated by a uniform grid, as shown in Figure 3. To test the safety of the nodes on the grid, the temporal velocity obstacle $NLVO(t_h)$ is computed. Nodes inside $NLVO(t_h)$, marked red in Figure 3, are marked inadmissible. Nodes out of $NLVO(t_h)$ are further evaluated by computing from each the unconstrained (no obstacles) minimum time-to-go (to the goal), as discussed

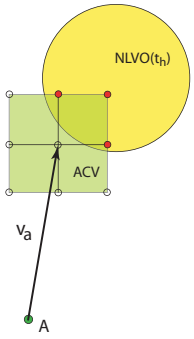


Fig. 3. Attainable Cartesian Velocities

next. The node with the lowest time is then explored to the next time step. This is repeated until reaching the goal. For one obstacle, this planner is guaranteed to reach the goal in the near minimum time. For many moving obstacles, it may not, and a global search may be required. Using only one temporal $NLVO(t_h)$ to determine potential collisions represents a significant computational gain, compared with the computation of the velocity obstacle for $t \in (0, t_h)$.

V. Examples

The on-line planner was implemented and tested for obstacle-free, and crowded static and dynamic environments. In the first example, shown in Figure 4, the robot, represented by a point, starts near point $(0.25, -1)$ at zero speed, attempting to reach the goal at point $(0.25, 2)$ (marked by a red triangle) at zero speed, while avoiding two obstacles, one static and one moving (to the right). The trajectory is shown in six snapshots, starting from the top left, and ending at the bottom right of Figure 4. In each snapshot, the two obstacles are shown in blue, together with their temporal velocity obstacles shown at the respective time horizon. Also shown is the robot trajectory up to that point from the start, with the velocity marked at the current point. Note that as the time horizon decreases, the size of the velocity obstacle increases (per (13)). At first, the robot turns left to avoid penetrating the velocity obstacle. This turn to the left occurs before the robot reaches the obstacle. After passing the static obstacle on the left, it turns right, to avoid the moving obstacle. At some point, the robot grazes the obstacle on the right, after which it is enclosed by the second velocity obstacle. This does not indicate a collision since its velocity points outside of the velocity obstacle. At that point, the relative velocity of the robot relative to the obstacle is tangent to the obstacle, as it should for the two to be sliding relative to each other. After avoiding the moving obstacle, the robot turns to the left to reach the goal.

The second example, in Figure 5, shows the robot avoiding 70 static obstacles. The robot accelerates and slows down through narrow passages toward the goal. Attempting to avoid the static obstacles with a too small time horizon resulted in the robot crashing early on into one of the obstacles.

The next example, in Figure 6, compares the local and

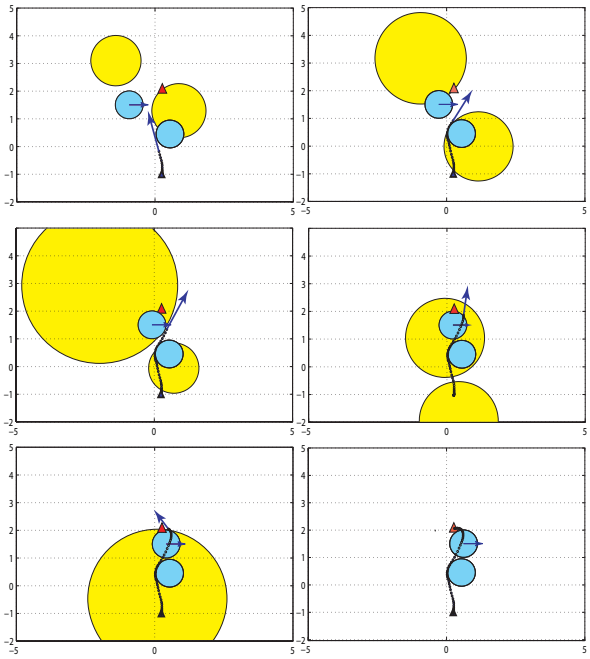


Fig. 4. Avoiding a static and a moving obstacle. Obstacles are shown in blue, and their respective velocity obstacles shown in yellow. The velocity vector is guided not to penetrate the velocity obstacles.

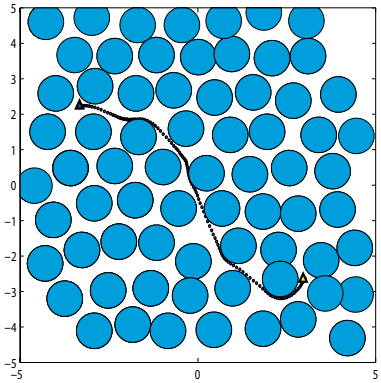


Fig. 5. Avoiding 70 static obstacles

the global planners in avoiding 70 static obstacles, starting from the bottom left. The local trajectory is shown in black, and the global trajectory in red. The two are almost identical until the global takes a left turn whereas the local takes a right turn before reaching the goal. The local trajectory was traveling 10% longer than the global solution.

The last example, in Figure 7, shows four snapshots of the robot avoiding 70 moving obstacles. It starts from the bottom center and moves to the target at the top right. A video clip of the full run is available in www.ariel.ac.il/me/pf/shiller/oren.

VI. Conclusions

An efficient mapping of obstacles to forbidden states for on-line planning in dynamic environments was presented. It consists of generating velocity obstacles at a carefully selected time horizon. This time horizon is selected for each

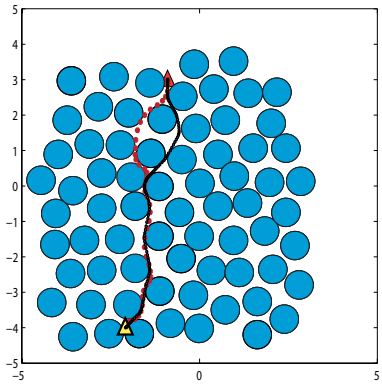


Fig. 6. Local (black) and global (red) trajectories avoiding 70 static obstacles

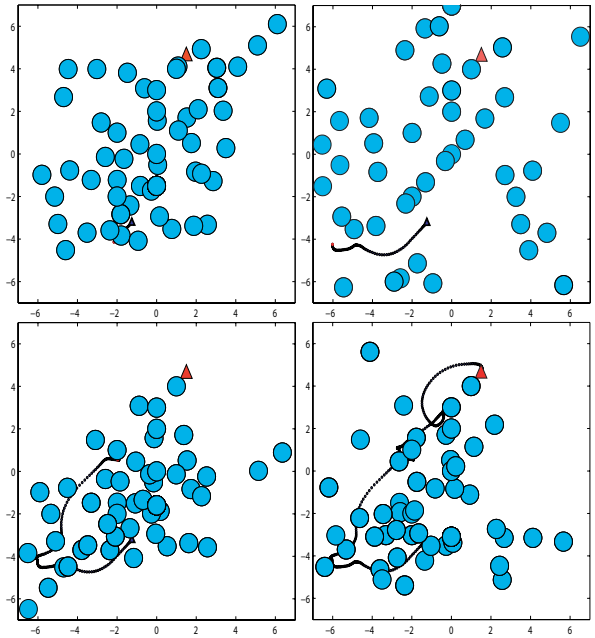


Fig. 7. Avoiding 70 moving obstacles

obstacle, static or moving, as the smallest of the minimum stopping and minimum passing times from the current state. Keeping the robot's velocity vector out of the velocity obstacle ensures that the robot does not enter unsafe states from which avoidance cannot be guaranteed. Recognizing unsafe states using the velocity obstacles is not only safe but also very efficient as it drastically reduces the search tree. The approach was demonstrated in an on-line planner that generates near time-optimal trajectories. The planner was demonstrated for a point mass dynamic model. Other robot models can be used with minor modifications. It is suitable for real time generation of high speed trajectories in crowded static and dynamic environments.

REFERENCES

[1] O. Brock and O. Khatib. Real time replanning in high- dimensional configuration spaces using sets of homotopic paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[2] J. Chan, N. Kuffner and M. Zucker. Improved motion planning speed and safety using region of inevitable collision. In *ROMANSY*, pages 103–114, July 2008.

[3] W. Fox, D. Burgard and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4:23–33, 1997.

[4] H. Fraichard, T. Asama. Inevitable collision state-a step towards safer robots? *Advanced Robotics*, 18:1001–1024, 2004.

[5] T. Fraichard. Planning in dynamic workspace: a state-time space approach. *Advanced Robotics*, 13:75–94, 1999.

[6] T. Fraichard. A short paper about safety. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[7] Hsu D. Kindel R. Latombe J-C. and Rock S. Randomized kinodynamic motion planning with moving obstacles. *Algorithmics and Computational Robotics*, 4:247–264, 2000.

[8] N.Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. In *International Conference on Intelligence Robots and Systems*, 1998.

[9] J. LaValle, S. Kuffner. Randomize kinodynamic planning. *International Journal of Robotics Research*, 20:378–400, 2001.

[10] Frazzoli E. Daleh M.A and Feron E. Real time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance Control and Dynamics*, 25:116–129, 2002.

[11] J. Minguez and L. Montano. Nearest diagram navigation. a new real-time collision avoidance approach. In *International Conference on Intelligence Robots and Systems*, 2000.

[12] J. Minguez, N. Montano, L. Simeon, and R. Alami. Global nearest diagram navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

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

[14] T. Petti, S. Fraichard. Safe motion planning in dynamic environment. In *International Conference on Intelligence Robots and Systems*, 2005.

[15] C. Stachniss and Burgard W. An integrated approach to goal-directed obstacles avoidance under dynamic constrains for dynamic environment. In *International Conference on Intelligence Robots and Systems*, 2002.

[16] L. Ulrich and J. Borenstien. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.

[17] M.S. Wikman, T.S. Branicky and W.S. Newman. Reflexive collision avoidance: a generalized approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1993.

Session IV

Motion planning

- **Title: Probabilistic Rapidly-exploring Random Trees for autonomous navigation among moving pedestrians**
  Authors: Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier

Probabilistic Rapidly-exploring Random Trees for autonomous navigation among moving obstacles.

Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier
LIG, INRIA Rhône-Alpes, France

***Abstract*— The paper presents a navigation algorithm for dynamic, uncertain environment. The static environment is unknown, while moving pedestrians are detected and tracked on-line. The planning algorithm is based on an extension of the Rapidly-exploring Random Tree algorithm, where the likelihood of the obstacles trajectory and the probability of collision is explicitly taken into account. The algorithm is used in a partial motion planner, and the probability of collision is updated in real-time according to the most recent estimation. Results show the performance for a car-like robot among a pedestrian tracking dataset and simulated navigation among multiple dynamic obstacles.**

I. INTRODUCTION

Autonomous navigation in populated environments represents still an important challenge for robotics research. The key of the problem is to guarantee safety for all the agents moving in the space (people, vehicles and the robot itself). In contrast with static or controlled environments, where path planning techniques are suitable [1] [2], high dynamic environments present many difficult issues: the detection and tracking of the moving obstacles, the prediction of the future state of the world and the on-line motion planning and navigation. The decision about motion must be related with the on-line perception of the world and take into account the sources of uncertainty involved:

1) The limits of the perception system: occluded zones, limited range, accuracy and sensibility, sensor faults;
2) The future behaviour of the moving agents: model error, unexpected changes of motion direction and velocity;
3) New agents entering the workspace;
4) Errors of the execution system.

Many real world applications rely on reactive strategies: the robot decides only about its immediate action with respect to the updated local estimation of the environment [3]–[5]. These strategies present however some major drawback: first of all the robot can be stuck in local minima; secondly, most of the developed approaches do not take into account the dynamic nature of the environment and the uncertainty of perception, so that the robot can be driven in dangerous or blocking situations.

To face these problems, reactive techniques are combined with global planning methods: a complete plan from present state to goal state is computed on the basis of the a priori information; during execution, the reactive algorithm adapts the trajectory in order to avoid moving and unexpected obstacles [6]–[8]. If the perception invalidates the planned path replanning is performed. In all the cited methods however, uncertainty is usually not taken explicitly into account.

From the more theoretical point of view instead, many works handle a non-deterministic or probabilistic representation of the information and the planning under uncertainty problem is solved using Markov Decision Processes (MDP), Partially Observable MDPs or game theory [9]–[11]. For an overview see [2]. These approaches are however very expensive from the computational perspective, and are limited to low dimensional problems or to off-line planning. In [12] and [13] a navigation strategy based on typical pattern based and probabilistic prediction is used in a planning algorithm based on a complete optimization method, A^* . However, the problem of A^* and of all complete methods is that the computational time depends on the environment structure and obstacles: these methods are more adapted to a low dynamic environments, where the information does not change frequently, the obstacles velocity is limited and the robot can stop often and plan its future movements. Also, they require a discretization of both the state and the control space, which reduces drastically the space for finding a feasible solution, expecially for robots with non-holonome or car-like constraints. Some recent work proposes to integrate uncertainty in randomised techniques, such as Probabilistic Road Maps [14] and Rapidly-exploring Random Trees (RRT) [15] [16].

In a highly dynamic environment an *anytime* algorithm is needed, which is able to give a feasible solution at ”anytime” it is asked to. In this paper we address the problem of taking explicitly into account the uncertainty in sensing and in prediction. We want our navigation algorithm to integrate new information coming from the perception system and to be able to react to the changes of the environment. In previous work [17] we developed a probabilistic extension of the RRT algorithm to handle a probabilistic representation of the static environment and of the moving obstacles prediction. The search algorithm has been integrated in a navigation algorithm which updates the probabilistic information and chooses the best partial path on the searched tree. The navigation algorithm is based on the architecture of Partial Motion Planning (PMP, [18]), where execution and local planning work in parallel to assure safe behaviour. The static environment is initially unknown and the robot explores it and builds an occupancy grid. While in [17] motion patterns were represented by Gaussian Processes, in this paper we consider two cases: in the first case the obstacles are simply tracked and their motion model is estimated on the basis of

previous observations; in the second case, the obstacles are supposed to follow pre-learned motion patterns which are represented by Markov chains and prediction is based on Hidden Markov Models.

The reminder of this paper is structured as follows: section III describes the representation of the static and dynamic world and how the probability of collision of a configuration is computed. Section IV recalls the RRT basic algorithm and details the new proposed approach. Section V recalls the PMP method and describes the planning and navigation algorithm developed. Results are presented in Section VI: an experiment with a laser scan dataset with moving pedestrians is presented and results in a simulated environment are shown. Section VII ends the paper with remarks and ideas for future work.

II. The Robot and the State Space

We consider a car-like robot moving in $\mathbb{R}^2$. The configuration space $\mathcal{C} = \{x, y, \theta, v, \omega\}$ described respectively by the position, orientation, linear and angular velocities of the robot in the workspace. The robot moves according to its motion model $q(t+1) = F(q(t), u(t))$ where input u is given by pairs (a, α) with a the linear and α the angular acceleration. The robot is subjected to kinematic and dynamic constraints: the linear velocity v is limited in the interval $[0, v_{max}]$ and the angular velocity ω is limited in $[-\omega_{max}, \omega_{max}]$. a and $alpha$ are also bounded: $a \in [a_{min}, a_{max}]$ and $\alpha \in [\alpha_{min}, \alpha_{max}]$.

Time is represented by the set $\mathcal{T} = (0, +\infty)$, which is the infinite set of discrete instants with measure unit the timestep τ. We define state space $\mathcal{X}$ of the robot, the space that represents the configuration of the robot at a certain instant in time $\mathcal{X} = \mathcal{C} \times T$. In the workspace there are static and moving obstacles. The task of the robot is to move from the initial configuration q_0 to a goal configuration q_{goal} in finite time without entering in collision with any obstacle. A solution trajectory is a sequence of states from q_0 to q_{goal} that is feasible according to the motion model of the robot and that is collision free: ie each configuration and each transition of the sequence are collision free. We assume that the position of the robot is known at each instant and that the robot moves following according to its motion model without error. In the deterministic case, the configuration space $\mathcal{C}$ can be divided in $\mathcal{C}_{free}$, the set of free configurations of the robot and $\mathcal{C}_{obs}$, the set of configurations where the robot is in collision with an obstacle. In our case instead we want to give a probabilistic representation of environment perception and prediction uncertainty and we need to define a probability of collision for each robot configuration. In the following paragraphs we explain how this probability is computed for a considered state of the robot X_r.

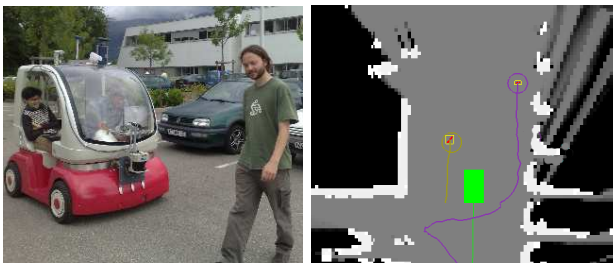
III. Probability of Collision

A. The Static environment

The 2D static environment is represented by an occupancy grid [19]: the space is divided in square cells. The environment is initially unknown, and the probability of occupation P_{occ} of each cell is fixed at 0.5. During navigation the space is observed by mean of a distance sensor (laser range finder). Assuming static environment, the probability of occupation of each cell is recursively updated according to the observations and estimated using a Bayesian filter. The probability of collision of a point in the space is given by the probability of occupation of the correspondent cell. For the set $S = (i, j)_N$ of N cells covered by the robot in state X_r, the probability of collision with static obstacles is given by the maximum probability over the set:

$$P(coll(X_r, \mathcal{G})) = \max_S(P_{occ}(i, j)) \quad (1)$$

Since the grid represents the static world, there is not need of prediction and the probability of collision does not depend on the time at which the robot is in a certain configuration.



(a) (b)

Fig. 1. (a) The cycab in the parking at INRIA Rhône-Alpes and (b) an occupancy grid with the robot (green rectangle) and 2 moving obstacles (coloured circles) along with their estimated trajectories.

B. Moving obstacles

Lets assume that the moving obstacles O_i can be approximated by circles of fixed radius. The state of an obstacle is $X = (x, y, \theta, v)$, its position in the 2D space, its orientation and linear velocity. Given an object observation Z, the belief state X and the prediction are estimated using Bayesian inference.

In a first case we will consider that the moving obstacles are detected and tracked by the robot using a Multi Hypothesis target Tracking (MHT) algorithm based on a set of Kalman Filters as in [20]: the motion of the obstacles can be represented with M linear motion models hypotheses A_m, each affected by zero-mean white Gaussian noise $\mathcal{N}(0, Q_m)$. At a considered instant t, the estimation of the state of an object is represented by a weighted sum of Gaussians (Gaussian mixture):

$$P(O_i(t)) \leftarrow \sum_{m=1}^{M} \alpha_m \cdot \mathcal{N}(\hat{X}_m(t), \Sigma_m(t)) \quad (2)$$

The prediction $\hat{X}$ can be analytically computed from the last estimation applying recursively the motion model. The obtained distribution is always a mixture of M Gaussians. Considering time horizon $t + k$:

$$\hat{X}_m(t+k) = A_m \cdot X_m(t+k-1) = A^k \cdot X(t) \quad (3)$$

$$\hat{\Sigma}(t+k) = (A_m^T)^k \cdot \Sigma_m(t) \cdot A_m^k + \sum_{j=0}^{k} (A_m^j \cdot Q_m) \quad (4)$$

Considering a state of the robot $X_r(t)$ and a moving obstacle O_i, the probability of collision is given by the integral of the probability distribution over the area S covered by the robot and enlarged by the radius of the obstacles:

$$P(coll(X_r, O_i)) = \iint_S P(coll(X_r, O_i)) = \\ = \sum_{m=1}^{M} \alpha_m \iint_S \mathcal{N}(\hat{X}_m^t, \Sigma_m^t) \quad (5)$$

The integral in previous equation is approximated sampling the distribution uniformly with the probability and considering the ratio between the number of samples inside and outside area S.

In a second case we will consider obstacles moving according to Hidden Markov Models as in [21]. The belief of the state at time t is given by a discretized distribution over the states of the Markov model. The prediction at time horizon $t+k$ is recursively estimated propagating the estimated state:

$$P(X(t+k)|Z(t)) = \\ \sum_{X(t+k-1)} P(X(t+k)|X(t+k-1))P(X(t+k-1)|Z(t))$$

where the first term in the sum is the probability to pass from state $X(t+k-1)$ to state $X(t+k)$ specified by the edges in the Markov model and the second is given by the observation model. The integral in Eq. 5 is here substituted by the sum over the states in the HMM touched by the area S.

Considering multiple moving obstacles, the total probability of collision is given by the probability of colliding with one OR another obstacle. Under the assumption that collisions with each obstacle are conditionally independent, the following equation is obtained:

$$P(coll(X_r, O)) = 1 - \prod_i (1 - P(coll(X_r, O_i)) \quad (6)$$

In the same way the probability of collision considering both the static environment and the moving obstacles is obtained:

$$P \quad (coll(X_r, O, \mathcal{G})) = \quad (7) \\ = \quad 1 - (1 - P(coll(X_r, \mathcal{G})) \cdot (1 - P(coll(X_r, O)))$$

C. New obstacles entering the scene

In dynamic environments, obstacles can enter or exit the workspace during the navigation task. Also if partial planning is used, it should be taken into account that new obstacles can enter the the workspace and interfere with the next motions of the robot. If it is possible to predict from where and when some obstacle may enter the scene, a more robust planning can be performed. The robot must:

- Distinguish from where a new obstacle may come.
- Apply a probability to the fact that an obstacle may enter and a motion model.

For the first problem the robot searches for specific areas from where an obstacle may enter (*doors*). This technique is based on some assumptions about the observed space and the size, shape and behaviour of the obstacles. The robot must be able to recognize on-line the *doors* with its perception only.

In the parking environment where we tested our algorithm, we assumed that obstacles may enter only traversing hidden areas: i.e. they cannot pass through static obstacles. Given a partial map and the point of view of the robot, these regions are easily extracted: the distance between the the points on the scan is studied and intervals bigger than the minimal size of an obstacles are kept as possible doors. For each interval, the partial map is observed to see if the area around is occupied, free or occluded. If the area is occupied, the considered interval is discarded: the area is occupied by a static obstacle that is hidden only from the current point of view. If the area is free or occluded ($P_{occ} \leq 0.5$), a door is recognized.

The probability of a new obstacle entering in the workspace can be modeled as an homogeneous Poisson process. The probability that at least one obstacle enters the scene, is given by the following equation:

$$P[N(t+\tau) - N(t) \geq 1] = 1 - e^{-\lambda\tau} \quad (8)$$

The rate parameter λ, is the expected number of arrivals per unit time. This parameter could be derived from a learning phase or fixed a priori. When performing prediction, an obstacle is initialized just behind the door, in the nearest possible point with respect to the robot actual position. The probability of occupation correspondent to the obstacle grows with the length of the time period of prediction according to Eq. 8. Using a worst-case hypothesis, obstacles are supposed to move toward the robot. A noise in both direction and a velocity is added to the model to take into account the other possibilities of motion.

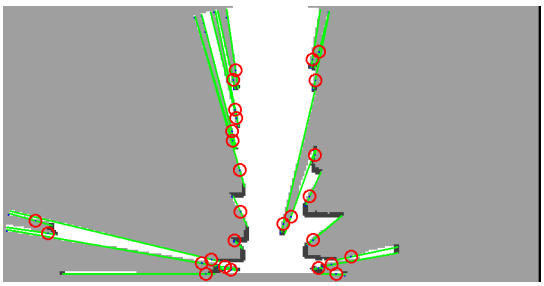
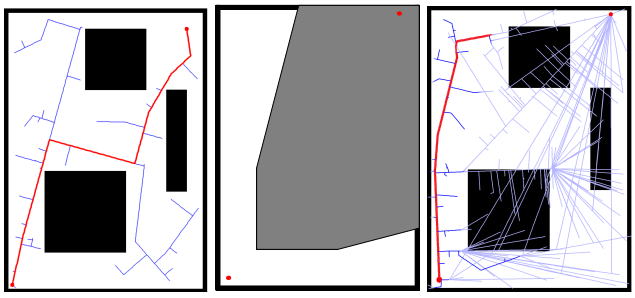


Fig. 2. A partial grid map, the extracted *doors* and the supposed new entering obstacles.

IV. Probabilistic RRTs

A. Basic Algorithm for RRTs

The Rapidly-exploring Random Tree (RRT) is a well known randomized algorithm to explore large state space in a relatively short time. The pseudocode of the algorithm is given in Algorithm 1. The algorithm chooses a point p in the state space and tries to extend the current search tree toward that point. p is chosen randomly, but in single-query planning, some bias toward the goal is generally applied in



(a) (b) (c)

Fig. 3. (a) RRT basic algorithm applied to a point holonome robot in a known static environment; (b) Perception given by a distance sensor at the initial position: white, black and grey represent respectively free, occupied and occluded zones; (c) Probabilistic RRT built in limited time: the search tree and the likelihood of the nodes in blue (lighter colour is for lower likelihood) and the chosen partial path in red.

Algorithm 1: Basic RRT.

```
   Data: T
1  while q_goal ∉ T do
2      p = ChoosePoint (q_goal);
3      q = T. NearestNeigbohr (p);
4      q_new = extend (q, p);
5      if q_new ∈ C_free then
6          T. addSon (q, q_new);
7  end
8  q = q_goal;
9  path = add (q);
10 while q ≠ T.root do
11     q = T. parentNode (q);
12     path = add (q);
13 end
```

order to speed up the exploration. p is chosen in the limited $\mathcal{C}_{free}$ (line 2). The nearest neighbour q of p within the nodes of the search tree is chosen for extension. A new node is obtained applying an admissible control from the chosen node q toward p (line 3). If q is collision-free, it is added to the tree. The algorithm can be stopped once the goal is found (line1) or it can continue to run to find a better path. The algorithm lies on a deterministic representation of the environment, so that both in the static and dynamic case we have a priori information on if a node is collision free or not and add it or not to the search tree. Once the goal state is reached, the path from the initial state to the goal is retrieved. Fig. 3(a) shows a point holonome robot in a known environment with static obstacles. The initial position of the robot is in the left corner at the bottom while the goal is in the upper right corner. An example of the search tree (blue lines) and the found path (red line) is shown; different running of the algorithm would give different results. In this case, the robot is supposed to move along straight lines, so that the Euclidean distance can be used to determine the nearest neighbour in the current tree. The algorithm can be generalized for car-like robots setting a different $\mathrm{NearestNeighbor}(.)$ function. and limiting the set of possible actions to the admissible controls of the robot from the node configuration.

B. Introducing probabilistic uncertainty

As stated in previous sections, the robot knowledge about the environment is incomplete in both space and time (sensor range, occlusions, new moving obstacles) and uncertain (sensor accuracy, motion model of the moving obstacles). On the basis of the RRT algorithm we developed an exploring algorithm which takes into account probabilistic uncertainty. For each configuration q of the space, a probability of collision $P_c(q)$ is computed considering the static and moving obstacles and the perception limits as in Eq. 7. The probability of reaching a particular configuration q_N is then given by the probability to cross the tree from the root q_0 to the considered node, i.e. the probability of *not* having collision in each of the traversed nodes:

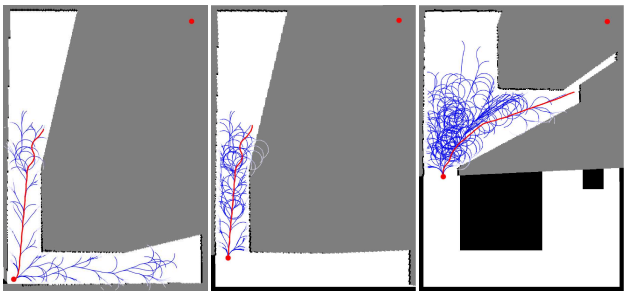
$$\begin{aligned} P_s(\pi(q_N)) &= P_s(q_0 ... q_N) \\ P_s(q_0 ... q_N) &= (1 - P_c(q_N)) \cdot P_s(q_0 ... q_{N-1}) \\ &= \prod_{n=0}^{N} (1 - P_c(q_n)) \end{aligned} \tag{9}$$

where we have considered that collision in subsequent nodes is statistically independent. We call this probability the probability of *success* P_s of the path. The probability falls exponentially with the length of the path. This is a sign that longer path are more dangerous, as the uncertainty accumulates over subsequent steps. All nodes than can be added to the tree, or a minimum threshold P_{smin1} can be chosen in order to avoid keeping in the tree very unlikely paths. Once a point p is chosen in the configuration space, the node to grow next q is chosen in dependence both on a measure of the expected length of the path $dist(q_0, q, p)$ and on the probability of success of the path. More precisely, $P_s(q_N)$ is normalized by the length N of the path and multiplied by the inverse of the distance to the chosen point.

$$\tilde{w}_q = \frac{1}{dist(q0, q, p)} \sqrt[N]{P_s(q)} \tag{10}$$

$$w_q = \frac{\tilde{w}_q}{\sum_q \tilde{w}_q} \tag{11}$$

The normalization is taken out so that the probability of success does not depend on the length of the path, which is taken instead into account by the distance term. The function $dist(q_0, q, p)$ is a sum of the length of the path from the root q_0 to the considered node and of the shortest path from q to p, which is a lower limit for the length of the eventual path to p. The obtained weights $\tilde{w}_q$ are normalized over the set of nodes in the tree (Eq. 11), and the result is a distribution over the nodes. The node to grow next is chosen taking the maximum or drawing a random node proportionally to the probability. In our implementation we choose the second case which appeared to be more robust to local minima. Even if a path to the goal is found, the algorithm can continue to search for a better/safer path, until a path is asked for execution. However, it is not guaranteed that a *safe enough* path is found even in infinite time, because of the environment uncertainty. The chosen path is then the best path that is safe enough, i.e. for which $P_s(q_N) \geq P_{smin2}$.



(a) (b) (c)

Fig. 4. Partial Probabilistic RRT applied to static environment for a point non-holonome robot. The tree updated and grown at three instants during navigation (in blue) and the chosen partial path (in red).

In general, this threshold is different from P_{smin1}: when the tree is updated and grown after new observations (see §sec:OnLineNavigation) the probability of each path is infact modified. Fig. 3(b) shows the perception given by a distance sensor in a static environment: areas behind the obstacles are unknown to the robot ($P_c \simeq 0.5$). Fig. 3(c) shows the tree grown by the described algorithm for an holonome point robot. The colour of the edges of the tree depends on the likelihood of the associated path: the lighter the colour the lower the likelihood. In red, the best path chosen.

V. On-line navigation

A. Related work: the Partial Motion Planning

In a dynamic environment the robot has a limited time to perform planning which depends on the time-validity of the models used and on the moving objects in the environment. The conditions used for planning could be invalidated at execution time: for example an obstacle could have changed its velocity or some new obstacle could have entered the scene. The idea of Partial Motion Planning [18] is to take explicitly into account the real-time constraint and to limit the time available for planning to a fixed interval. After each planning cycle, the planned trajectory is generally just a partial trajectory. The exploring tree is updated with the new model of the world and the final state of the previous trajectory becomes the root of the new exploring tree. The planning algorithm works in parallel with execution. Each node of the tree is guaranteed to be not an Inevitable Collision State (ICS, [22]) by checking if it exists a collision free braking trajectory from the node. This is a conservative approximation that does not allow the robot to pass an intersection before an approaching moving obstacle. Our approach presents an adaptable time horizon for planning. The time for the planning iterations depends on the length of the previous computed trajectory and on the on-line observations. Safety of a path is guaranteed studying braking trajectories only for the last state of the path.

B. Developed Algorithm

When the robot moves, it observes the environment and updates its estimation with the incoming observations. The cost of crossing the tree changes and the tree needs to be updated. The update consists in three steps:

1) Prune the tree: the new root is the position of the robot and nodes that are in the past are deleted; the probability of reaching the nodes is updated, taking into account that the robot has already crossed part of the tree.
2) Update the weight of the nodes: when a change in the probability of collision is detected, the weight of the correspondent nodes (and of their subtree) must be updated.
3) Retrieve the best path.

If the considered environment is dynamic we need the robot to do these operations in real-time. In better words we need to know how much time is available for updating and how to allocate it. In the first step, the present state of the robot is considered. The tree is pruned so that only the subtree attached to the state of the robot is maintained. When the probability to pass from a configuration q_0 to q_i changes, the weight of the subtree attached to q_i is updated using the following equations:

$$P(q_N|q_i) = (P(q_N|q_0) - \hat{P}(q_i|q_0))\frac{1}{1-\hat{P}(q_i|q_0)} \quad (12)$$

$$P(q_N|q_0) = P(q_i|q_0) + (1 - P(q_i|q_0))P(q_N|q_i) \quad (13)$$

The first equation gives the probability of traversing the tree from q_i to q_N, assuming that the probability of reaching q_i changed from $\hat{P}(q_i|q_0))$ to 1: q_0 is the old root, q_i is the new root and q_N is a node in the family of q_i. This first update is used when the tree is pruned and is due to the fact that the robot has already moved from q_0 to q_i, so that the new $P(q_i|q_0)$ is 1. The second equation gives the probability to traverse the tree from q_0 to q_N when the probability to pass from q_0 to q_i changes from 1 to $P(q_i|q_0)$. Eq. 12 and 13 are used one after the other when the observations revealed some difference with the prediction. In this case q_0 and q_i are respectively the start and ending configuration in which a change in the probability of collision has been detected. In Fig. 4, the on-line updating of the tree is shown at 3 instants during navigation. At the beginning, the most likely paths are explored in the two possible directions and the most promising one is chosen. Fig. 4(b) shows the tree after some steps: the tree has been updated: the branch in the right direction has been cut has is not reachable anymore and the tree has been grown. Fig. 4(c) shows the tree and the new partial path found when a bigger portion of the space is visible

VI. Experimental Results

The planning algorithm has been tested with real data acquired on the car-like vehicle (Cycab) shown in Fig. 1(a). To test the algorithm we define a goal 20 meters ahead the robot at each observation cycle and let the algorithm run in parallel with the online mapping and tracking (Fig. 6(b)). The planning algorithm runs at 2Hz. An example of the grown tree and the chosen path is shown in Fig. 5. The occupancy

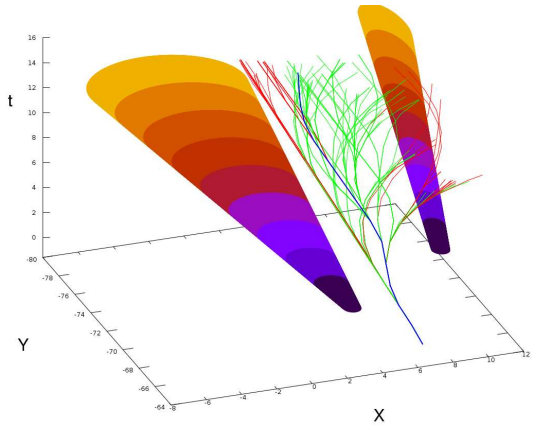


Fig. 5. The prediction of the moving obstacles and the explored tree in (x, y, t) space.

grid correspondent to the figure is the one in Fig. 1(b). The two cones represent the prediction of the two moving pedestrians considering ellipses of axes correspondent to one standard deviation interval. A threshold has been applied to show different colours for safer (green) and dangerous (red) paths. The best path is shown in blue. Each sequence is then tested with the real data, letting a virtual robot move through the map. Fig. 6(a) shows the observed occupancy grid (a) and the tree of states explored in the available time (b): lighter blue is for higher probability of collision. The red line is the chosen path. Fig. 6(c-f) shows subsequent positions of the virtual robot; on the background the predicted occupancy grid at the correspondent planning stage, while the red circles represent the real position of the obstacles at the considered time. Results prove that the algorithm is able to compute safe trajectories in real time taking into account the static environment, the moving obstacles perceived and their velocity and the uncertainty which arises from a real dataset.

The navigation strategy has been tested in the Cycab simulator (7(a)). A rectangular environment has been simulated. A certain number of doors is simulated for the two long sides of the rectangle. Obstacles are supposed to enter from a door and to exit by another door in the opposite side. The space has been discretized in a uniform cell grid of step $0.5m$ An 4-connected HMM graph has been built on the grid for each goal: the probability to pass from a state to another depends on the decrease of the distance to the goal between the origin state and the destination one. A certain amount of noise is applied so that states that present nearly the same decrease in distance are given the same probability. The probability is then normalized over the set of edges coming out from the origin node. A set of trajectories has been randomly simulated on the basis of the graph: for each trajectory the enter door and the exit door are chosen (Fig. 7(b)). Given a state of the obstacle, the next state is drawn proportionally with the edges probability. The position of the obstacle inside the cell is chosen by a smoothing filter.

The simulated robot has the same dimensions and kinematic and dynamic constraints of the Cycab. Perception is assumed

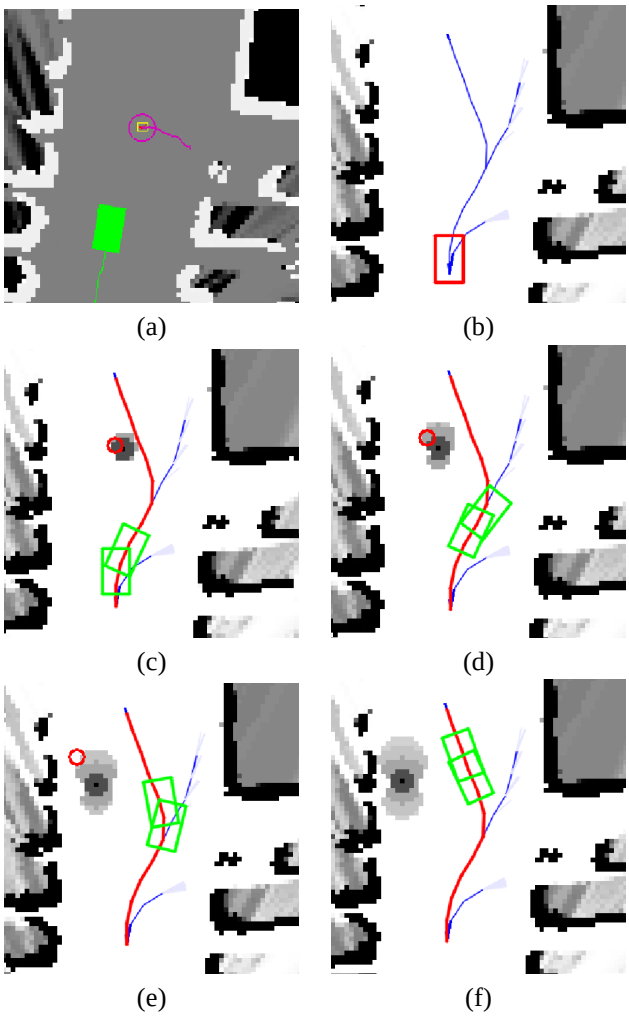


Fig. 6. Planning results with a laser dataset. (a) The static environment is mapped and the moving obstacles are tracked. (b) The algorithm explores the state space and chooses a path. (c-f) the path is compared with the prediction and the real observations acquired.

perfect: the obstacles are represented by circles of $0.30m$ radius whose position is always known; no occlusion or finite range is considered. The robot has to cross the environment and successively reach goals which are positioned randomly in the environment, with some bounds near the walls. The robot knows where the doors are and the Markov graph correspondent to the simulated trajectories. Prediction is performed on the basis of Hidden Markov Models, as in [**?**] or [21] and the probability of collision is computed sampling the obtained distribution on the cells. Fig. 7(c-f) show the robot (green rectangle) traversing the environment to reach the goal: the red line is the partial path computed at the time-step in the shot, while red circles represent the moving obstacles with their previous trajectory attached. The robot reached 1000 goals with various numbers of pedestrians simulated in the space. No collision with the robot in motion was detected during the experiment, while the number of collisions as 0 velocity grows with the number of objects in the space. To understand these results, we must notice that the simulated obstacles do not have any knowledge of the

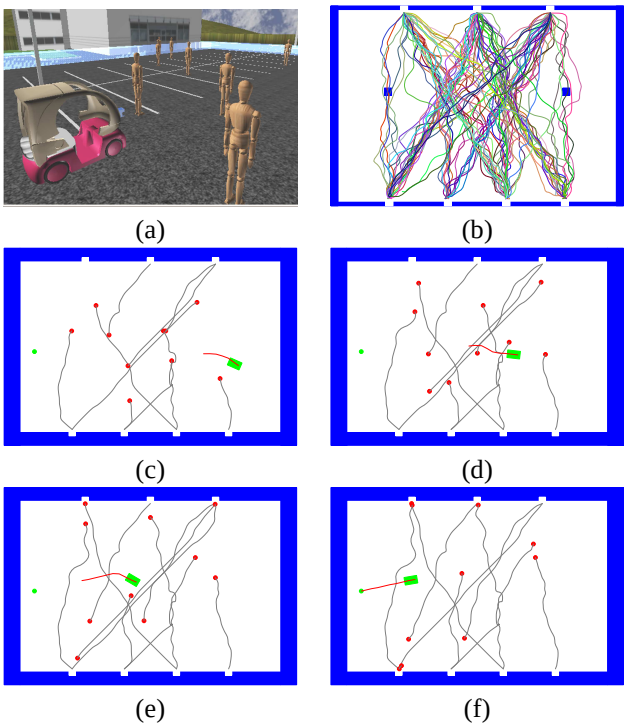


Fig. 7. Navigation results in simulated environment. (a) The Cycab simulator with the robot and simulated pedestrians. (b) The simulated trajectory dataset. (c-f) Navigation among moving pedestrians based on HMM probabilistic prediction.

robot and that its kinematic possibilities are strongly limited if compared to those of the obstacles: as the robot cannot go backward, it tends to avoid obstacles and get stacked with the walls of the environment, while the obstacles continue to move around it.

VII. Conclusion and future work

The paper presents a navigation algorithm which integrates perception uncertainty and incompleteness in the planning strategy using a probabilistic framework. The tests prove that the robot is able to navigate in real-time reacting properly to unexpected changes of the environment and reaching the given goal positions. The use of an adaptable time horizon for planning makes the algorithm both reactive to unexpected changes of the environment and *forward looking* when previously planned trajectories are not invalidated by observation.

Immediate work will deal with testing the navigation algorithm to have a measure of its performance in more complex and realistic scenarios. Future work will deal with the integration of the localization and execution uncertainty in the planning algorithm and with testing the navigation with the real robot.

References

[1] J. C. Latombe, *Robot Motion Planning*. Dordrecht, The Netherlands: Kluwer, 1991, vol. SECS 0124.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[3] J. Borenstein and Y. Koren, "The vector field histogram - fastobstacle avoidance for mobile robot," *IEEE Transaction on Robotics and Automation*, vol. 7, no. 3, June 1991.

[4] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *IEEE International Conference on Robotics and Automation, ICRA*, 1996.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, Mar. 1997.

[6] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *IEEE International Conference on Robotics and Automation, ICRA*, 1993.

[7] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *IEEE International Conference on Intelligent Robots and Systems, IROS*, 2002.

[8] F. Large, "Navigation autonome d'un robot mobile en environnement dynamique et incertain," Ph.D. dissertation, Université de Savoie, 2003.

[9] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '95)*, July 1995, pp. 1080 – 1087.

[10] S. LaValle and R. Sharma, "On motion planning in changing, partially-predictable environments," *Int'l J. Robotics Research*, vol. 16, pp. 775–805, 1997.

[11] A. Foka and P. Trahanias, "Real-time hierarchical pomdps for autonomous robot navigation," *Robot. Auton. Syst.*, vol. 55, no. 7, pp. 561–571, 2007.

[12] M. Bennewitz and W. Burgard, "Adapting navigation strategies using motion patterns of people," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*. IEEE, 2003, pp. 2000–2005.

[13] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research (IJRR)*, vol. 24, no. 1, 2005.

[14] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Proc. of IEEE International Conference on Robotics and Automation*, 2006, pp. 1261–1267.

[15] R. Benenson, S. Petti, M. Parent, and T. Fraichard, "Integrating perception and planning for autonomous navigation of urban vehicles," in *IEEE IROS*, 2006.

[16] N. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *IEEE ICRA*, April 2007.

[17] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1056–1062, Sept. 2008.

[18] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *IEEE IROS*, 2005.

[19] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[20] J. Burlet, "Adaptive interactive multiple models applied on pedestrian tracking in car parks," 2008, phD thesis.

[21] D. Vasquez, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, INP de Grenoble, February 2007. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/Vas07

[22] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *IEEE IROS*, 2003.