

Integrating Perception and Planning for Autonomous Navigation of Urban Vehicles

Rodrigo Benenson, Stéphane Petti
INRIA Rocquencourt / Mines Paris
78153 Le Chesnay Cedex, France
firstname.lastname@inria.fr

Thierry Fraichard
INRIA Rhône-Alpes
38334 St Ismier Cedex, France
thierry.fraichard@inria.fr

Michel Parent
INRIA Rocquencourt
78153 Le Chesnay Cedex, France
michel.parent@inria.fr

Abstract—The paper addresses the problem of autonomous navigation of a car-like robot evolving in an urban environment. Such an environment exhibits a heterogeneous geometry and is cluttered with moving obstacles. Furthermore, in this context, motion safety is a critical issue. The proposed approach to the problem lies in the coupling of two crucial robotic capabilities, namely perception and planning. The main contributions of this work are the development and integration of these modules into one single application, considering explicitly the constraints related to the environment and the system.

I. INTRODUCTION

In many urban environments, private automobile use has led to severe problems with respect to congestion, pollution, and safety. A large effort has been put in industrial countries into developing new types of transportation systems, the Cybercars are an answer to this problem [15]. Cybercars are city vehicles with fully automated driving capabilities. Such autonomous systems cannot be realized without using several capabilities designed to work together in a single application. To safely navigate, the system will have to model the environment while localizing in it, plan its trajectory to the goal and finally execute it. The problem of designing and integrating such capabilities, while accounting for the various constraints of such an application, remains largely open and lies at the heart of the work presented in this paper.

Autonomy in general and motion autonomy in particular have been a long standing issue in Robotics. Several architectures have been proposed. They mainly differ in the context as well as the robotic platform which is intended to perform the task. At first the environment imposes its own constraints. Indeed, within an urban environment, moving objects (eg pedestrians, other cars, etc.) imposes a real time constraint on the navigation scheme which is the time that the system has to take a decision. When a robot is placed in a dynamic environment, it cannot stand still, otherwise it might be hit by a moving object. Besides, in a dynamic environment, the future motion of the moving obstacles is usually not known in advance and will have to be predicted. Since the urban environment is partially predictable, it is possible to provide a valid prediction over a limited time horizon. At second, a complex system as a car-like robot is constrained by its (nonholonomic) kinematics as well as its dynamics. It is therefore of the utmost importance to explicitly account for

these different constraints in order to safely move the robot to its goal.

Most of the work on autonomous vehicles has been applied to simple indoor robots for which kinematic and dynamic constraints are usually not considered. Furthermore, they usually rely on strong geometric assumptions for the map construction, and disregard the moving obstacles. Some interesting autonomous navigation systems considering moving obstacles and relaxed geometric constraints were presented by [25] and more recently by [14]. In the last years significant advances have provided medium to high speed autonomous vehicles evolving outdoors [1], [10]. These systems are able to evolve in structured and non structured environment, considering the dynamic constraints of the vehicle and the presence of static obstacles. Recently an autonomous navigation architecture integrating moving obstacles and safety notions was presented [17]. However they rely on a structured environment assumption and do not explicitly integrate the dynamic environment considerations at the planning stage. Finally some previous works have discussed the safety issues in urban environments and their relation to the perception requirements [22].

Previous approaches differ in several ways, however it is clear that an autonomous robot placed in a partially predictable dynamic environment must have perceptive, deliberative and reactive capabilities. In this paper, the perception relies on a Simultaneous Localization and Mapping (SLAM) algorithm extended for moving objects detection and tracking so as to build a world model including static obstacles as well as a short term prediction of the moving obstacles motions. The deliberative scheme uses these model to generate trajectories that explicitly account for the constraints of the environment and the system. The approach which is used relies on a deliberative strategy that interleaves planning with execution. It consists in incrementally and iteratively calculating a safe trajectory to the goal in order to provide motion autonomy to the system. To the authors' knowledge, the approach presented in this paper is the first to handle explicitly the dynamic nature of the environment and the kinematics and dynamics of the system.

We detail in §II the perception algorithm and in §III the planning scheme. In §IV we present the integration of both modules and the results of experiments performed on a real car-like robot. Finally we draw some conclusions and discuss

the future work in §V.

II. PERCEPTION IN URBAN ENVIRONMENTS

A. Introduction

Perception is the process of transforming measures of the world into an internal model. The kind of model (and the choice of the sensors) depends on the application. For autonomous navigation, the world model needs to integrate at least four elements: the target to attain, the position of the static obstacles, the current and future position of moving obstacles and the current state (position, speed, etc.) of the vehicle.

Due to occlusion and limited field of view the robot can not observe the entire world at each measurement. Integrating successive observations into a consistent map of forward obstacles is required to create an effective planning. It is well known that it exists a duality between creating consistent maps and localizing the robot. Such duality has been extensively studied as the Simultaneous Localization And Mapping (SLAM) problem [24].

Unfortunately most of the works in SLAM suppose that the environment is static. The presence of moving obstacles will contaminate the map and perturb the data association between two observations. For the planning purpose we require to explicitly identify the moving obstacles and estimate they current state in order to predict they future position.

We can see that for autonomous navigation, as a strict minimum the robot requires to solve the Simultaneous Localization, Mapping and Moving Objects Tracking (SLAMMOT) problem [27]. In the following paragraphs we will propose a solution to this problem and then we will discuss the additional considerations required when integrating perception and planning.

The key point to create correct maps (and thus correctly localize the robot) is to successfully do data association between current and past measures.

Data association methods have a limited “attraction region”, if the initial guess is outside this region the association will produce an erroneous result. The attraction region depends of the existing map, the current measure and the method employed.

When the robot successfully recognize a previously visited place the SLAM algorithms will allow to reduce its pose uncertainty helping thus in the data association process.

Due to space limitation we will not discuss the city sized SLAM problem. The Incremental Maximum Likelihood method [24] is a simple approach for small scale map construction. The incurred error is acceptable when the robot does not close a loop and the drift inside the map is under the desired bound. The incremental construction of the map eliminates the need to store the previous measures or to recompute online the map. A set of small scale maps can be used as building blocks for a larger map.

In outdoor mobile robotics, the sensors commonly employed to observe the surrounds are video cameras, radars and laser scans [23]. We choose the last one due of its larger range

(more than 180° and 40 meters) and high precision ($\pm 1^\circ$ and ± 0.1 meters). Notice that the laser scanner measures provide information about the presence of obstacles and the existence of free space.

B. Laser scan data association

Laser scan data association (so called “scan matching”) can be used both to estimate small displacements between two measures, and to recognize a revisited place.

The classic method for scan matching (both in 2D and 3D) is the Iterative Closest Point (ICP) [29]. This iterative method is straightforward but provide slow convergence rates and low attraction regions. This is why many variants have been proposed [19], changing the point to point association methods or changing the optimization metrics [13].

Recently a new approach has been proposed [6], [8]. Instead of matching two cloud of points, a cloud of points is matched over a distribution of probabilities indicating the probable presence of an object at each point of the space. This approach has the advantages of allowing error modeling of the sensor, avoiding the expensive closest point search and providing more robust results with faster convergence rates.

One method of this family, called Normal Distribution Transform (NDT) has been successfully applied to robotic applications [2]. This method can be seen as crude but fast approximation for modelling the space occupancy probability distribution, or as an enhanced version of the traditional occupancy grid representation [7].

Instead of approximating the occupancy probability by a grid of squares, it is approximated by a grid of overlapping gaussians distributions. The space is subdivided in a grid, and each cell is associated to one or more gaussian distributions [2]. When a new point hit a cell the associated gaussians parameters are incrementally updated. Since the gaussians approximate locally the observed obstacles, the representation is much finer than the grid granularity (see fig. 1).

Each bi-dimensional Gaussian is defined by its mean vector q and its covariance matrix Σ . A laser scan measure is defined as a set of point x_i . Then the score function between a match the occupancy distribution can be written as:

$$score = \sum_i \exp \left(-\frac{1}{2} \cdot (x'_i - q_i)^T \Sigma_i^{-1} (x'_i - q_i) \right) \quad (1)$$

Where x'_i is the scan point x_i in the map reference frame (translated using current pose estimate) and q_i , Σ_i are respectively the mean and covariance of a Gaussian covering the point x'_i (can be more than one gaussian per point).

The objective of scan matching is to search the displacement of the scan that optimize the score of (1). The derivatives of the score function can be written explicitly and are cheap to evaluate. Thus optimization algorithms such as gradient descent and Newton’s can be applied directly. It has been experimentally validated that this approach is faster and more robust than ICP [12].

Since the grid of gaussians can be updated incrementally, it does not only provide a good scan matching method, but it can also be used as a map representation.

The second derivatives of the score function can be written explicitly, so the Hessian matrix can be evaluated at the computed optimum point. Then this matrix can be used to approximate the uncertainty of the scan matching. This is very useful for a good estimation of the pose uncertainty, the ICP algorithm and its variants do not provide such a cheap way to do this [27, chapter 3].

In the experiments presented here, we are not yet dealing with the revisiting problem (a core aspect of the SLAM problem). However since the presented data association is more robust, it is at least more adequate than plain ICP. If more computing time is available it is possible to enhance the matching method using stochastic search or with a multiresolution extension [6], [18].

In the next subsection we will discuss how to merge the grid of gaussians representation with a moving objects detection method.

C. Moving objects detection and tracking

Many works discuss how to detect moving objects, or how to construct maps of static objects. However little work has been done in doing both simultaneously. The proposed methods include offline optimization [3], [7, chapter 4] and online heuristics [7], [14, chapter 3]. First works on online SLAMMOT proposed to detect moving objects using a data consistency approach between successive laser scans [27]. This approach was then formalized in a bayesian estimation formulation using a modified grid of occupancy [28]. Here we will discuss how to integrate this last method with a grid of gaussians representation.

The core notion to detect moving objects is the inconsistencies between observed free space and observed occupied space. If free space appears where a static object was observed, then it probably moved. If measures appear in areas previously seen as free, then this measures probably correspond to moving objects.

Let be $P(S_t^x)$ the static obstacle occupancy probability at the point x and the instant t . Instead of updating the occupancy probability $P(S_t^x)$ using only the last observation value o_t , the update depends both of the observation value o_t and of the last occupancy estimate $P(S_{t-1}^x)$.

The probability of occupancy is divided in three ranges: Free, Unknown and Occupied. Then the relation $P(S_t^x | S_{t-1}^x, o_t)$ enforcing the coherence between free and occupied space observations can be illustrated as shown in table I. The case when the last observation gives no information about the occupancy probability, $P(S^x | o_t) = Unknown$, is omitted.

The occupancy probability update is then written as

$$\begin{aligned} odds(x) &= P(x)/(1 - P(x)), \\ odds(S_t^x | o_{1..t}, S_{1..t-1}^x) &= \\ odds(S_t^x | o_t, S_{1..t-1}^x) \cdot odds(S^x)^{-1} \cdot odds(S_{t-1}^x). \end{aligned}$$

TABLE I
INVERSE OBSERVATION MODEL FOR THE STATIC OCCUPANCY
PROBABILITY [28].

$P(S_{t-1}^x)$	$P(S^x o_t)$	$P(S_t^x S_{t-1}^x, o_t)$
Free	Free	Low
Unknown	Free	Low
Occupied	Free	Low
Free	Occupied	Low
Unknown	Occupied	High
Occupied	Occupied	High

In order to merge this approach with the grid of gaussians representation we propose to separate the storage of occupancy measures O_{occ} and the free space measures O_{free} .

$$\begin{aligned} O_{occ} &= \{o | P(S^x | o) = Occupied \text{ and } o \in o_{1..t}\} \\ O_{free} &= \{o | P(S^x | o) = Free \text{ and } o \in o_{1..t}\} \end{aligned}$$

Since $odds(S_t^x | o_{1..t}, S_{1..t-1}^x)$ is estimated from a multiplication series, this series can be divided and reduced to two separate factors. One factor $odds_{occ}^x$ accounts for the occupancy estimation based on occupied space measures and the second factor $odds_{free}^x$ accounts for the occupancy estimation based on free space measures.

$$\begin{aligned} odds_{occ}^x &= odds(S_t^x | O_{occ}, S_{1..t-1}^x) \\ odds_{free}^x &= odds(S_t^x | O_{free}, S_{1..t-1}^x) \end{aligned}$$

Then occupancy probability can be retrieved at any moment multiplying the two values.

$$odds(S_t^x | o_{1..t}, S_{1..t-1}^x) = odds_{free}^x \cdot odds_{occ}^x$$

Doing this separation the grid of gaussians can be used directly. If points are added to a gaussian only when $P(S_{t-1}^x) = Occupied$ then the gaussian distribution evaluated at x can be used as an approximation for $odds_{occ}^x$.

In order to clean the gaussians that correspond to a space that is no more occupied it is necessary to keep an estimate of the occupancy probability at its mean value q_i (we suppose that the shift of mean point during gaussians parameters updates does not invalidate the occupancy probability estimate). When $P(S_t^{q_i}) = Free$ the corresponding gaussian is erased.

The factor $odds_{free}^x$ can be estimated using any representation (including coarse or fine grids). In our implementation we use a bi-linear interpolation between the corners of a cell of the grid of gaussians. An illustration of the resulting occupancy probability scalar field can be seen at figure 1.

The proposed method still being a gross approximation (just as grid methods), however separating occupancy and free area factors allow to better control the approximation used. More precise approaches would consider updating the gaussian parameters when portions of it pass to free regions. The proposed approach use a lightweight representation that allows fast matching and the detection of moving objects.

At the end of the scan matching, each point x'_i has already been evaluated over its corresponding gaussians, thus $odds_{occ}^{x'_i}$ is available. Computing the $odds_{free}^{x'_i}$ allows to estimate $P(S_t^{x'_i})$.

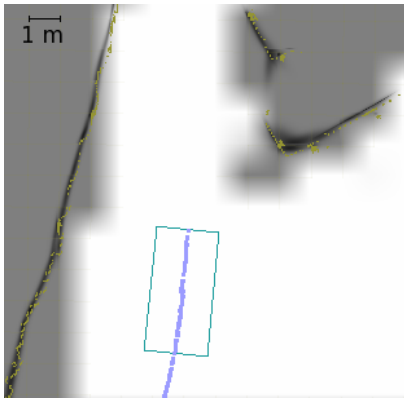


Fig. 1. Static occupancy probability scalar field approximation using a grid of gaussians and bi-linear interpolation. Cells size is 1 [m]. A vehicle and its past trajectory are also shown.

Points were $P(S_t^{c_i}) = Free$ are considered as moving objects measures.

Once we are able to detect moving objects we need to track them in order to estimate their state and predict their behaviour (since the prediction will be used for the planning stage). Tracking multiple moving objects is a classical problem. In the general case this problem is very hard, however it has been shown experimentally that simple methods are good enough to cope with urban scenarios [5], [27]. We use a similar approach than [5].

D. Safety considerations

In the driverless vehicle context, safety is associated to collision free trajectories. Since the world model provided by the perception module is the only information available for planning, we have to ensure that the trajectories without collisions generated in the predicted world will remain free of collisions during their realization in the real world.

To ensure this the world model need to do *consistent predictions*: predicted free space has to be effectively free in the real world future.

The future observations of the moving obstacles need to be inside the predicted occupied area. Integrating adequately the model error into the predictions allows to have consistent predictions. However, too loose predictions (large models errors) will generate large banned areas forcing the planning to be too much conservative.

In order to have a consistent prediction, we do not only have to deal with the measured moving obstacles, but also with not yet observed ones. At the unobserved limits of the field of view frontier we have to assume the possible appearance of moving obstacles. To ensure trajectories free of collisions, we need to suppose the worst case, i.e. the presence of obstacles moving directly toward the current robot position at the maximum expected speed. Creating such virtual obstacles will force the planning module to generate a trajectory conservative enough to deal with the sudden apparition of new obstacles.

In urban environment, the expected maximum speed of surrounding obstacles depends on their position. It would be

interesting to be able to model their maximum speed as a function of the space in order to make worst case estimations less conservative [26].

Once we are able to create a consistent world model in real time, we now need to construct a trajectory that respects both safety and computation time constraints.

III. PLANNING IN DYNAMIC ENVIRONMENT

A. Introduction

Planning in an environment cluttered with moving obstacles implies to plan under a real time constraint. Indeed, a robotic system placed in a dynamic environment has a limited time only to compute the motion plan to be executed. If the execution of the plan could begin at an arbitrary time, there would not be any problem. This is however not the case. In a real dynamic environment, a robotic system cannot safely remain passive as it might be collided by a moving obstacle. This time the system has to make its decision is the *decision time constraint*, δ_d and is therefore a real-time constraint imposed by the environment.

Early work addressing the problem of navigation within dynamic environments, rely on *reactive* approaches. These methods consist in a local exploration of the velocity space, i.e. the set of all possible velocities of the robot, in order to find the proper velocity to be applied during the next time step. For robots controlled in speed and steering angle, the velocity output can be directly executed by the robot, which makes these techniques particularly efficient. Their local nature exhibit however strong limitations in terms of convergence. Besides, complex kinematic or dynamic constraints are difficult to handle in a general way, without resorting to crude approximations. Recently, *deliberative* methods accounting for time constraints, have been also presented. Deliberative methods, also referred to as motion planning methods, consist in calculating a priori a complete motion plan to the goal. Some approaches based on improved dynamic programming techniques, have been presented [11]. These methods however are restricted to low dimension problems and cannot account for general kinematic or dynamic system's constraints. Recent random techniques have been presented with very fast and impressive results for higher dimension problems [9]. The real time constraint is however never explicitly considered and therefore no computation time upper bound can be guaranteed. Due to the complexity of the motion planning problem, sometimes referred to as "the curse of dimensionality", there is little hope that within an arbitrary bounded time, a complete plan to the goal might be found. Therefore, the proposed approach to the problem is a Partial Motion Planner (PMP) that guarantees a bounded computation time at the expense of its completeness, i.e. the guarantee to plan a complete trajectory to the goal.

B. Notations

Let \mathcal{A} denote the car-like robot placed in a workspace \mathcal{W} (fig. 2). The model of the car-like robot used in the planning

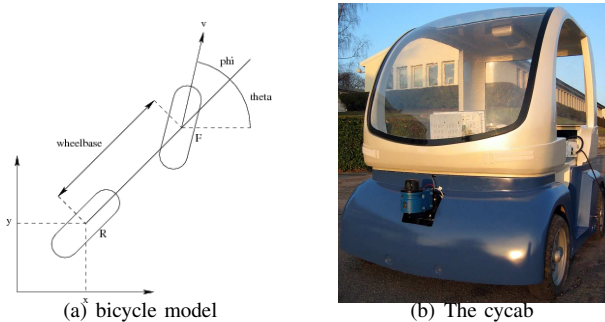


Fig. 2. The car-like vehicle. \mathcal{A} .

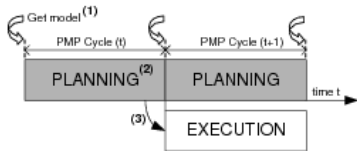


Fig. 3. Partial Motion Planning architecture.

strategy is described by the following differential equation :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v_r \cos \theta \\ v_r \sin \theta \\ v_r \frac{\tan \phi}{L} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma \quad (2)$$

This equation is of the form $\dot{s} = f(s, u)$ where $s \in \mathcal{S}$ is the state of the system, \dot{s} its time derivative and $u \in \mathcal{U}$ a control. \mathcal{S} is the state space and \mathcal{U} the control space of \mathcal{A} . A state of \mathcal{A} is defined by the 5-tuple $s = (x, y, \theta, v, \xi)$ where (x, y) are the coordinates of the rear wheel, θ is the main orientation of \mathcal{A} , v is the linear velocity of the rear wheel, and ξ is the orientation of the front wheels. A control of \mathcal{A} is defined by the couple (α, γ) where α is the rear wheel linear acceleration and γ the steering velocity, with $\alpha \in [\alpha_{min}, \alpha_{max}]$ (acceleration bounds), $\gamma \in [\gamma_{min}, \gamma_{max}]$ (steering velocity bounds), and $|\xi| \leq \xi_{max}$ (steering angle bounds). L is the wheelbase of \mathcal{A} , $\mathcal{A}(s)$ is the subset of \mathcal{W} occupied by \mathcal{A} at a state s . Let $\phi \in \Phi: [t_0, t_f] \mapsto \mathcal{U}$ denote a control input, *i.e.* a time-sequence of controls. Starting from an initial state s_0 , at time t_0 , and under the action of a control input ϕ , the state of the system \mathcal{A} at time t is denoted by $s(t) = \phi(s_0, t)$. An initial state and a control input define a trajectory for \mathcal{A} , *i.e.* a time sequence of states.

C. Partial Motion Planner (PMP) Algorithm

The partial motion planner (PMP) is a motion planning strategy that explicitly accounts for the real time constraint imposed by the environment. Besides, in a real environment, the model of the future can be predicted over a limited time only δ_v . Therefore, PMP is structured around a constant planning cycle (PMP cycle in fig. 3) of duration δ_c , in order to be able to regularly get an update of the model. This

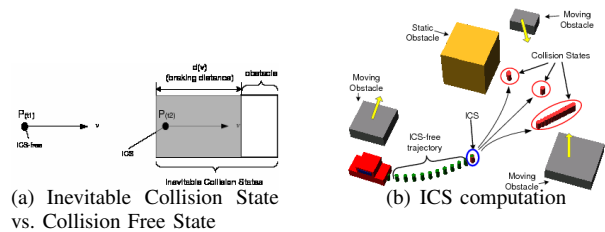


Fig. 4. Inevitable Collision States (ICS).

cycle duration must in fact fulfill the requirement that $\delta_c = \min(\delta_d, \frac{1}{2}\delta_v)$. The main cycle of PMP is described as follows, starting at time t_i :

- 1) Get an updated model of the future.
- 2) The state-time space of \mathcal{A} is searched using an incremental exploration method that builds a tree rooted at the state $s(t_{i+1})$ with $t_{i+1} = t_i + \delta_c$.
- 3) At time t_{i+1} , the current iteration is over, the best partial trajectory ϕ_i in the tree is selected according to given criteria (safety, metric) and is fed to the robot that will execute it from now on. ϕ_i is defined over $[t_{i+1}, t_{i+1} + \delta_{h_i}]$ with δ_{h_i} the trajectory duration.

After completion of a planning cycle, the planned trajectory of time horizon δ_h is most likely a partial trajectory. Thus, the PMP algorithm iterates over a new cycle of duration δ_c , as depicted in figure 3, until the goal is reached. The algorithm operates until the robot reaches a neighbourhood of the goal state. In case the planned trajectory has a duration $\delta_h < \delta_c$, the cycle of PMP can be set to this new lower bound or the navigation (safely) stopped. In practice however, the magnitude of δ_h is much higher than δ_c .

In our work, we use a sampling based incremental method. Sampling based methods avoid the complete space representation by probing the space by mean of a collision detection module. In our approach however, the usual geometric collision checker is replaced with an inevitable collision state checker described in the next part. This original module allows to deterministically extend the tree to the goal while insuring avoidance of static and dynamic obstacles. This method is incremental in order to be interrupted at any time. The control space of our system is reduced to the set of bang bang controls $\tilde{\mathcal{U}} = (\alpha, \gamma)$ with $\alpha \in [\alpha_{min}, 0, \alpha_{max}]$ and $\gamma \in [\gamma_{max}, 0, \gamma_{min}]$. The exploration of the state-time space consists in building incrementally a tree as follows. The closest state s_c to the goal is selected. A control from $\tilde{\mathcal{U}}$ is applied to the system during a fixed time (integration step). In case the new state s_n of the system is safe, this control is valid. The operation is repeated over all control inputs and finally the new state, safe and closest to the goal, is finally selected and added to the tree.

D. Safety Issues

Like every method that computes partial motion only, PMP has to face a safety issue: since PMP has no control over the duration of the partial trajectory that is computed, what

guarantee do we have that \mathcal{A} will never end up in a critical situations yielding an inevitable collision? We need however to define the safety we consider. In figure 4(a) we consider a selected milestone of a point mass robot P with non zero velocity moving to the right (a state of P is therefore characterised by its position (x, y) and its speed v). Depending upon its state there is a region of states for which P, even though it is not in collision, will not have the time to brake and avoid the collision with the obstacle. As per [4], it is an Inevitable Collision State (ICS). In this paper, we refer to a safe state as ICS-free.

In general, computing ICS for a given system is an intricate problem since it requires to consider the set of all the possible future trajectories. To compute in practice the ICS for a system such as \mathcal{A} , it is taken advantage of the approximation property established in [4]. This property shows that a conservative approximation of the ICS can be obtained by considering only a finite subset \mathcal{I} of the whole set of possible future trajectories. For our application we consider the subset \mathcal{I} of braking trajectories obtained by applying respectively constant controls $(\alpha_{min}, \gamma_{max})$, $(\alpha_{min}, 0)$, $(\alpha_{min}, \gamma_{min})$ until the system has stopped. Once it is still, it is checked to be collision free (*i.e.* over a trajectory obtained by applying constant (0,0) controls) until the end of the PMP cycle. In the PMP algorithm, every new state is similarly checked to be an ICS or not over \mathcal{I} . In case all trajectories are in collision, this state is an ICS and is not selected (see fig. 4(b)).

A safe trajectory consists of safe states. However, a practical problem appears when safety has to be checked for the continuous sequence of states defining the trajectory. In order to solve this problem and further reduce the complexity of the PMP algorithm, we presented in [16] a property that simplifies the safety checking for a trajectory. This property is important since first, it proves a trajectory is continuously safe while the states safety is verified discretely only, and second it permits a practical computation of safe trajectories by integrating a dynamic collision detection module within existing incremental exploration algorithms, like A* or Rapidly-Exploring Random Tree (RRT).

One difficulty when performing motion planning using an incremental approach is the choice of the metric used to select and expand the nodes in order to build the tree. This parameter is recognised to have a large influence on the trajectory quality specially when dealing with non-holonomic systems. In this work, the non holonomic continuous curvature (CC) metric presented in [21] is used. It greatly improves the convergence and quality of the planned trajectories compared to holonomic metrics.

IV. EXPERIMENTAL RESULTS

The algorithms presented in §II and §III, where implemented in C++ and integrated in an automated electric vehicle, the Cycab. Both algorithms are designed to incrementally and iteratively construct a solution which enables an efficient and simple interweaving. The tracking of the generated trajectories is insured by a non-linear closed loop controller detailed in

[20]. The integrated system is able to autonomously drive in real world environments toward goals lying within about a hundred meters, while avoiding static and moving obstacles. The complete software runs at 10 [Hz] on a standard 3.3 [GHz] PC. Currently the only input data used is one layer of an IbeoML laser scanner.

In figure 5 we present the result of an early experiment. The top pictures are snapshots of the world model constructed during a single experiment. Darker areas represent higher occupancy probability of static obstacles. Moving obstacles are represented by a circle. Current results do not include the estimation of unobserved obstacles. The dark rectangle describes the current vehicle pose and the light one, the desired vehicle pose (speeds are not shown). The executed trajectory is behind the vehicle and the current planned partial trajectory is represented in front of it. The bottom pictures show the corresponding scenes in the real world. During initial validation, the maximum speed of the Cycab is limited to low speeds (1.5 [m/s]), full speed experiments at higher speed (4 [m/s]) will be done in the future.

First results indicate that this new architecture is functional and provides the expected behaviour.

The large circles present in the map correspond to a semi-transparent fence. Interpreted as a moving obstacle with zero speed this perturbation does not affect the system behaviour.

V. CONCLUSION AND FUTURE WORKS

In this paper, we analyze the main difficulties associated to navigation in urban environments and propose a perception-planning duo able to cope with an heterogeneous environment populated by static and moving obstacles.

The perception algorithm provides a better data representation, coupled with faster data association and the detection of moving obstacles. The planning algorithm generates safe trajectories, in bounded time. Their successful integration provides for the first time an experimental validation of the proposal.

As future work, the perception module could be enhanced through a coupling with sidewalk detection, the use of collaborative perception architectures and the use of vehicles internal sensors. In particular we plan to extend the mapping method to city scale maps construction. Finally, we could add a high level road planner in order to build a city-scale system.

REFERENCES

- [1] <http://www.darpagrandchallenge.com>, 2006.
- [2] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003.
- [3] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in dynamic environments with mobile robots. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.
- [4] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
- [5] Kay Ch. Fuerstenberg, Dirk T. Linzmeier, and Klaus C.J. Dietmayer. Pedestrian recognition and tracking of vehicles using a vehicle based multilayer laserscanner. In *Proceedings of the 10th World Congress on Intelligent Transport Systems (ITS)*, Madrid, Spain, November 2003.

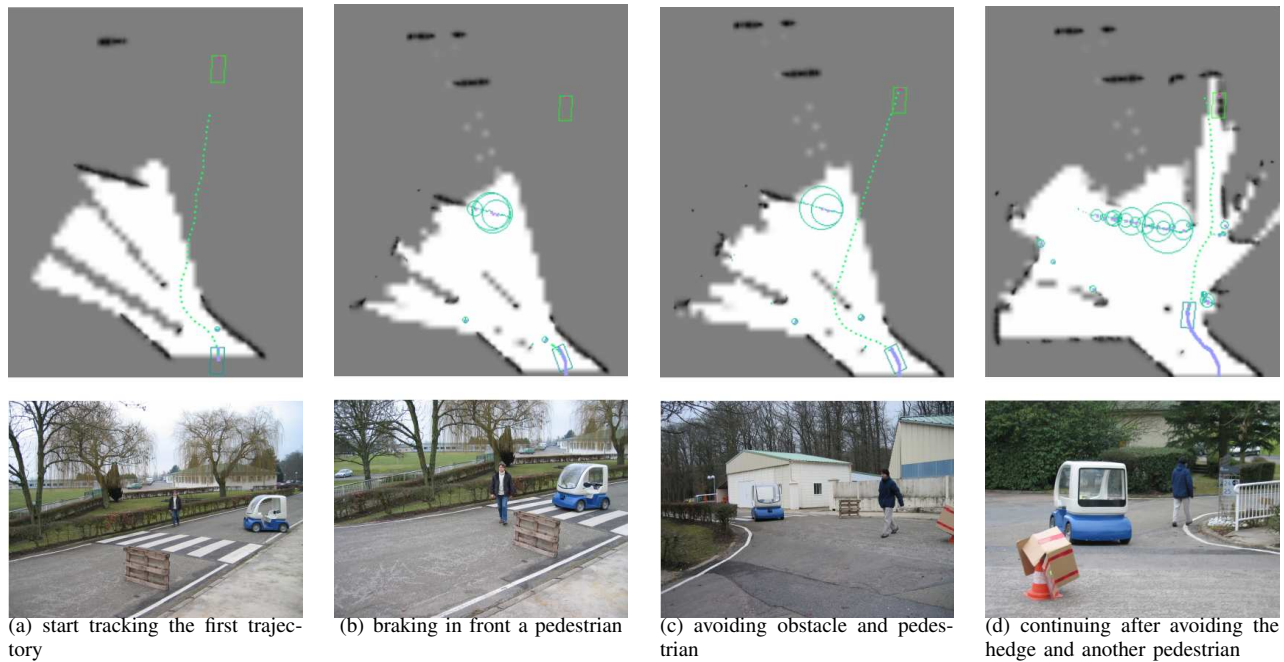


Fig. 5. Experiment results.

- [6] Sébastien Granger and Xavier Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision (ECCV 2002)*, volume 2353 of *LNCS*, pages 418–432, Copenhagen, Denmark, 2002. Springer.
- [7] Dirk Haehnel. *Mapping with Mobile Robots*. PhD thesis, Fakultät für Angewandte Wissenschaften, Universität Freiburg, December 2004.
- [8] Arie Kaufman Haitao Zhang, Olaf Hall-Holt. Range image registration via probability field. In *Computer Graphics International Conference*, pages 546 – 552, 2004.
- [9] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, March 2002.
- [10] S. Kolski, D. Ferguson, M. Bellino, and R. Seigwart. Autonomous driving in structured and unstructured environments. In *IEEE Intelligent Vehicles Symposium*, 2006.
- [11] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.
- [12] Martin Magnusson, Tom Duckett, Rolf Elsrud, and Lars-Erik Skagerlund. 3d modelling for underground mining vehicles. In *SimSafe 2005, Proceedings of the Conference on Modeling and Simulation for Public Safety*, 2005.
- [13] J. Minguéz, F. Lamiroux, and L. Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *Int. Conf. on Robotics and Automation*, Barcelona, Spain, 2005.
- [14] L. Montesano, J. Minguéz, and L. Montano. Modeling the static and the dynamic parts of the environment to improve sensor-based navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.
- [15] M. Parent. Automated public vehicles : A first step towards the automated highway. In *4th World Congress on Intelligent Transport Systems*, October 1997.
- [16] S. Petti and Th. Fraichard. Safe motion planning in dynamic environments. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB (CA), August 2005.
- [17] C. Pradaliér, J. Hermosillo, C. Koike, C. Braillon, P. Bessièrre, and C. Laugier. The cycab: a car-like robot navigating autonomously and safely among pedestrians. *Robotics and Autonomous Systems*, 50(1):51–68, 2005.
- [18] N. Ripperda and C. Brenner. Marker-free registration of terrestrial laser scans using the normal distribution transform. In *Proceedings of the ISPRS Working Group V/4 Workshop 3D-ARCH*, Mestre-Venice, Italy, August 2005.
- [19] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Int. Conf. on 3D Digital Imaging and Modeling*, 2001.
- [20] P. Rives S. Benhimane, E. Malis and J. R. Azinheira. Vision-based control for car platooning using homography decomposition. In *IEEE International Conference on Robotics and Automation*, pages 2173–2178, Barcelona, Spain, April 2005.
- [21] A. Scheuer and T. Fraichard. Planning continuous-curvature paths for car-like robots. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1304–1311, Osaka, Japan, November 1996.
- [22] C. Thorpe, J. D. Carlson, D. Duggins, J. Gowdy, R. MacLachlan, C. Mertz, A. Suppe, and C.C. Wang. Safe robot driving in cluttered environments. In *Int. Symposium of Robotics Research*, October 2003.
- [23] C. Thorpe, O. Clatz, D. Duggins, J. Gowdy, R. MacLachlan, J. R. Miller, C. Mertz, M. Siegel, C.C. Wang, and T. Yata. Dependable perception for robots. In *Int. Advanced Robotics Programme IEEE*, Seoul, Korea, May 2001. Robotics and Automation Society.
- [24] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.
- [25] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot. In *Int. Conf. on Robotics and Automation (ICRA)*, 1999.
- [26] A. D. Vasquez and Th. Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), April 2004.
- [27] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.
- [28] Denis F. Wolf and Gaurav S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, July 2005.
- [29] Zhengyou Zhang. Iterative point matching for registration of free-form curves. Technical Report RR-1658, INRIA-Sophia Antipolis, April 1992. Equipe : ROBOTVIS.