Towards urban driverless vehicles

Rodrigo Benenson, Stéphane Petti, Thierry Fraichard, Michel Null Parent

Toward urban driverless vehicles

Rodrigo Benenson, Stéphane Petti, Thierry Fraichard and Michel Parent

INRIA Rocquencourt,
Domaine de Voluceau. BP 105
78153 Le Chesnay Cedex, France
E-mail: {firstname.lastname}@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 an 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 design of perception and planning modules that consider explicitely the dynamic nature of the vehicle and the environment while enforcing the safety constraint. The main contributions of this work are the development of such modules and they integration into a single application. Initial full scale experiments validating the approach are presented.

Keywords: Driverless car, Cybercars, Motion planning, Perception, NDT, SLAM, MOT, PMP.

1 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, as an answer to this problem [19]. 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. Indeed, 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 has been a long standing issue in Robotics. Several architecture have been proposed. They mainly differ in the context as well as the platform which is intended to perform this task. At first the environment imposes its own constraints. Indeed, within an urban environment, its dynamic nature (pedestrians, other cars, etc...) imposes on the navigation scheme a real time constraint over the time that the system has to take a decision.

Figure 1 ParkShuttle system within Schipol airport parking lot.

In particular, when a robot is placed in a dynamic environment, it cannot remain still, otherwise it might be hit by a moving obstacle. 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. The intent of our work is to explicitly account for these different constraints in order to safely move the robot to its goal.

The first automated vehicles are designed to follow a predefined path. The path following problem, classic in control theory, requires the vehicle to estimate its instantaneous position with respect to the path. A dedicated infrastructure is used to materialize the path and enable the vehicle to relocalize on it. Early wire guided vehicles follow a wire buried in the ground, by mean of inductive sensors. Later techniques use dead-reckoning associated with relocalisation on magnets or transponders widely spaced in the ground. These techniques allow for fine tuning the exact path of the vehicles. On these configurations the safety is related to blocking the access of pedestrians to the special roads, or detecting obstacles on the way and stopping the vehicle. This kind of system have been already commercialized, such as on the ParkShuttle (Fig. 1).

Recent techniques permit the localisation of the vehicle using the natural features in the environment. These techniques have the advantage of requiring no modification of the environment. In this case, a collision free path must be computed which requires a model of the environment that can be built by extracting higher-level features. Autonomous vehicles that are infrastrucuture independent become thus very complex. Therefore, 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 environment model construction, and disregard the moving obstacles. Some interesting autonomous navigation systems considering moving obstacles and relaxed geometric constraints were presented by [29] and more recently by [18, 21]. In the last years significant advances have provided medium to high speed autonomous vehicles evolving in outdoor [30, 13]. 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 [22]. 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 [26].

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 an Simultaneously Localization and Mapping algorithm (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 this world model to generate trajectories that explicitly account for the constraints of the environment and the system. The approach which is used rely 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 integrated system that handles explicitly the dynamic nature of the environment and the kinematics and dynamics of the system.

We review some previous work on perception and planning in the following sections. At first in §2, where the proposed perception algorithm is detailed. At second, in §3, after reviewing the issues and related work, we present the planning scheme. In §4 and §5 we present the integration of both modules and the results of experiments performed on a real car-like robot, the Cycab. Future works are discussed in §6 and finally we draw some conclusions in §7.

2 Perception in urban environments

2.1 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 of the vehicle (position, speed, etc...).

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 know 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 [28].

Unfortunately most of the works in SLAM suppose a static environment. 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 their current state in order to predict their 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 [32]. 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 on the existing map, the initial estimate, 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.

The Incremental Maximum Likelihood method [28] is a simple approach for small scale maps 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 section §6 we discuss the city sized SLAM problem.

In outdoor mobile robotics, the sensors commonly employed to observe the surrounds are video cameras, radars and laser scans [27]. We choose the last one due to its larger range (more than 180° and 40 meters) and high precision ($\pm 1°$ and ± 0.1 meters). Notice that the laser scanner measures provide information about the presence of obstacles and the existence of free space. We are supposing that urban environments are locally planar and thus a 2D representation of the environment is good enough for navigation purposes. Since the world model needs to describe the presence of obstacles, dense measures of the environment are required. Also it has been observed that urban environments present a high variability in geometric shapes and little assumptions can be made about the expected patterns to appear [32, chapter 3]. Thus feature based data association, a common approach in indoor robotics, seems brittle for the urban environment application. Instead, a dense scan matching algorithm is proposed.

In subsection §2.2 we briefly describe the choiced method to associate successive measures in order to construct the local maps. Then in subsection §2.3 we explain how to detect measures of moving objects based on the consistency of the observed free and occupied space.

2.2 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) [35]. This iterative method is very straightforward but provides slow convergence rates and small attraction regions. This is why many variants have been proposed [24], changing the point to point association methods or changing the optimization metrics [16].

Recently a new approach has been proposed [8, 11]. 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 [1]. This method can be seen as a crude but fast approximation for modelling the space occupancy probability distribution, or as an enhanced version of the traditional occupancy grid representation [9]. Instead of approximating the occupancy probability by a grid of squares, it is approximated by a grid of overlapping gaussian distributions. The space is subdivided in a grid and each cell is associated to one or more gaussians distributions[1]. 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. 2).

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 scan and the occupancy distribution can be written as equation 1.

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

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

The objective of scan matching is to search the displacement of the scan that optimizes the score of (1). The derivatives of the score function can be written explicitly and are cheap to evaluate thus optimization methods such as gradient descent and Newton's can be applied directly. It has been experimentally validated that this approach is both faster and more robust than ICP [15].

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. Since the gaussians are estimated using measures from scans matched using the maximum aposteriori probability, they variance matrix will naturally represent the measure error. Using an regular grid make the assumption that the measure error still low (smaller than the grid granularity) as distance augment, which is true for a laser scanner. When using vision as the base sensor the error augment exponentially with the distance thus the proposed approach would be inadequate. In such case a multiresolution grid approach should be explored [17].

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 any cheap way to do this [32, chapter 3].

In the experiments here presented we are not yet dealing 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 [8, 23].

Table 1 Inverse observation model for the static occupancy probability [34].

$P(S^x_{t-1})$	$P(S^x\mid o_t)$	$P(S^x_t\mid S^x_{t-1},\ o_t)$
Free	Free	Low
Unknown	Free	Low
Occupied	Free	Low
Free	Occupied	Low
Unknown	Occupied	High
Occupied	Occupied	High

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

2.3 Moving objects detection and tracking

Many works discuss how to detect moving objects, and many others how to construct maps of static objects. However yet little work have been done in doing both simultaneously. The proposed methods include offline optimization [2, 9, chapter 4] and online heuristics [9, 18, chapter 3]. First works on online SLAMMOT proposed to detect moving objects using a data consistency approach between successive laser scans [32]. This approach was then formalized in a bayesian estimation formulation using a modified grid of occupancy [34]. 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 these measures probably correspond to moving objects.

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

The probability of occupancy is divided in three ranges Free, Unknown and Occupied. Then the relation $P(S^x_t\mid S^x_{t-1},\ o_t)$ enforcing the coherence between free and occupied space observations can be illustrated as shown in table 1. The case when the last observation gives no information about the occupancy probablity, $P(S^x\mid o_t) = Unknown$, is omitted. The distribution $P(S^x\mid o_t)$ is constructed based on the sensor characteristics, while $P(S^x_t\mid S^x_{t-1},\ o_t)$ is a design variable, see [34] for more details.

The occupancy probability update is then written as in equation 2.

$$\begin{aligned} &odds(x) = P(x)/(1-P(x)),\\ &odds(S^x_t\mid o_{1\dots t},\ S^x_{1\dots t-1}) =\\ &\quad odds(S^x_t\mid o_t,\ S^x_{1\dots t-1})\cdot odds(S^x)^{-1}\cdot odds(S^x_{t-1}). \end{aligned} \tag{2}$$

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}, as defined in 3.

$$\begin{aligned} O_{occ} &= \{o\mid P(S^x\mid o) = Occupied \text{ and } o\in o_{1\dots t}\}\\ O_{free} &= \{o\mid P(S^x\mid o) = Free \quad \text{ and } o\in o_{1\dots t}\} \end{aligned} \tag{3}$$

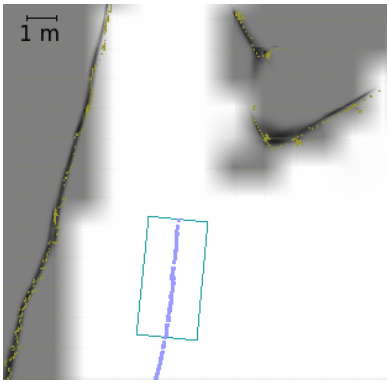


Figure 2 Static occupancy probability scalar field approximation using a grid of gaussians and bi-linear interpolation. Cells size is 1 [m]. A vehicle and its past trajectory are also shown.

Since $odds(S_t^x|o_{1...t},\ S_{1...t-1}^x)$ is estimated from a multiplication series (equation 2), this series can be split and reduced in two separate factors, defined at (4). The factor $odds_{occ}^x$accounts the occupancy estimation based in occupied space measures and the second factor $odds_{free}^x$ accounts the occupancy estimation based in free space measures.

$$\begin{aligned} odds_{occ}^x &= odds(S_t^x|\ O_{occ},\ S_{1...t-1}^x) \\ odds_{free}^x &= odds(S_t^x|\ O_{free},\ S_{1...t-1}^x) \end{aligned} \tag{4}$$

Then occupancy probability can be retrieved at any moment multiplying the two values, as shown by the equation 5.

$$odds(S_t^x|\ o_{1...t},\ S_{1...t-1}^x) = odds_{free}^x \cdot odds_{occ}^x \tag{5}$$

Doing this separation the grid of gaussians can be used directly as part of the detection of static obstacles and moving obstacles.

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 $odds_{occ}^x$. This means that the grid of gaussians provides an estimate of the static obstacles in the environment. Because of the dynamic nature of the environment, a previously static object can start moving. In order to clean the gaussians that correspond to 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 odd_{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 2.

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

allows to have a lightweight representation that allows fast matching and detection of moving objects.

At the end of the scan matching, each point x'_i has already been evaluated over its corresponding gaussians, thus $odd_{occ}^{x'_i}$ is available. Computing the $odds_{free}^{x'_i}$ allows to estimate $P(S_t^{x'_i})$. Points were $P(S_t^{x'_i}) = Free$ 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 be shown experimentally that simple methods are good enough to cope with urban scenarios [7, 32]. We use a similar approach to [7].

2.4 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 the planning we have to ensure that the trajectories without collisions generated in the predicted world, will still being 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 [31, 33].

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.

3 Planning in dynamic environment

3.1 Introduction

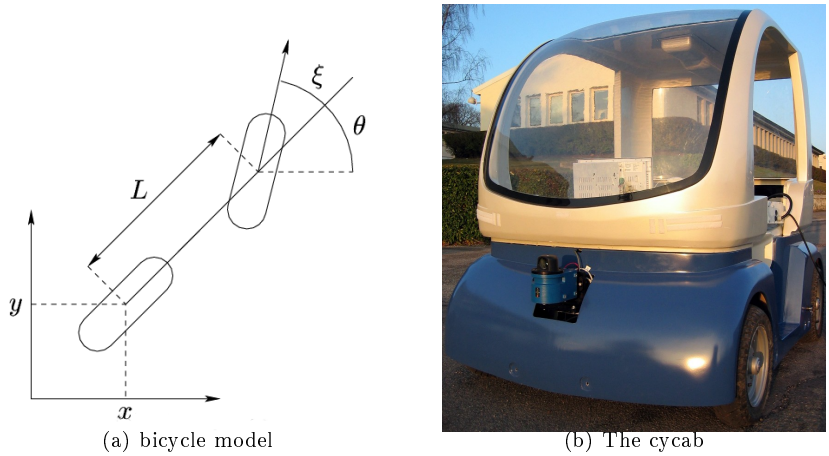
Planning in an environment cluttered with moving obstacles implies to plan under a real time constraint. Indeed, a robotic system placed in a dynamic environment has a limited time only to compute the motion plan to be executed. If the execution of the plan could begin at an arbitrary time, there would 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 [14]. 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 [12]. 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.

3.2 Notations

Let $\mathcal{A}$ denote the car-like robot placed in a workspace $\mathcal{W}$ (fig. 3). The model of the car-like robot used in the planning strategy is described by the following differential equation :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\frac{\tan\phi}{L} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma \tag{6}$$



(a) bicycle model (b) The cycab

Figure 3 The car-like vehicle. $\mathcal{A}$.

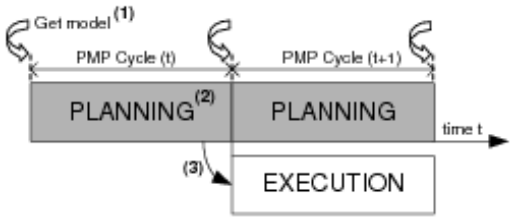


Figure 4 Partial Motion Planning architecture.

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 $|v| \leq v_{max}$(velocity bound), $\alpha \in [\alpha_{min}, \alpha_{max}]$ (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 $\phi \in \Phi$: $[t_0, t_f] \longmapsto \mathcal{U}$ 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.

3.3 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

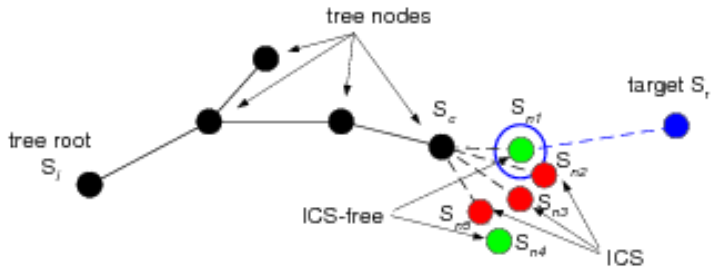


Figure 5 Tree construction over a PMP cycle.

fig. 4) of duration δ_c, in order to be able to regularly get an update of the model. This cycle duration must in fact fulfill the requirement that $\delta_c = min(\delta_d, \frac{1}{2}\delta_v)$. A typical PMP cycle 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 i, it is most likely the planned trajectory of time horizon δ_{h_i} is partial. Thus, the PMP algorithm iterates over a cycle of duration δ_c, as depicted in figure 4, 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 $\delta_{h_i} < \delta_c$, 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_i} is much higher than δ_c.

3.4 Space Exploration

In our work, we use an incremental sampling based method. Sampling based methods avoid the complete space representation by probing the space by mean of a collision detection module. Incremental methods have the advantage to be interuptible, *ie.* anytime the calculation is stopped, a partial answer can be returned. The exploration of the state-time space $\mathcal{ST}$ consists in building incrementally a tree by integrating the system over which a constant control is applied, over a small duration. To this mean, the control space of the system is discretized. We use the set of bang bang controls $\tilde{\mathcal{U}}$=(α, γ) with $\alpha \in [\alpha_{min}, 0, \alpha_{max}]$ and $\gamma \in [\gamma_{max}, 0, \gamma_{min}]$. The tree construction consists in the following (Fig. 5) :

1. the state s_c from the existing tree and closest to the target s_r is selected.
2. a control from $\tilde{\mathcal{U}}$ is applied to the system during the fixed integration step from s_c.

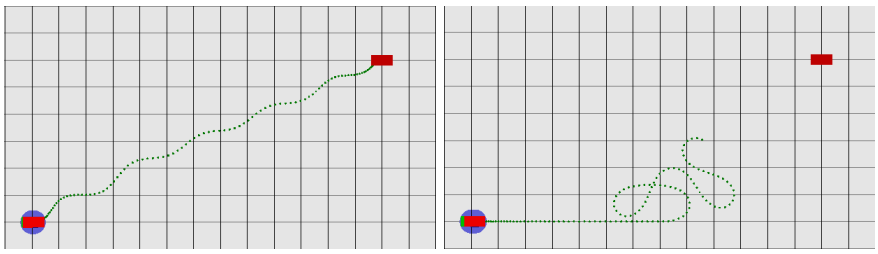
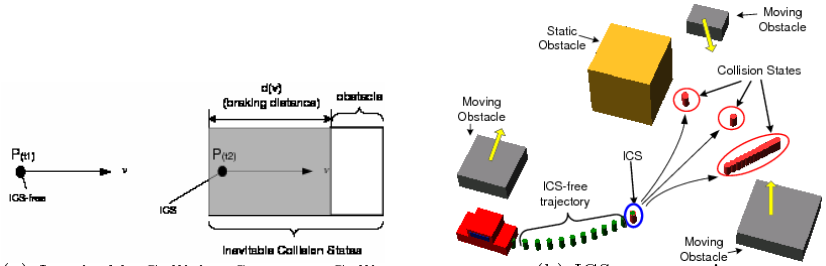


Figure 6 Influence of the metric on the generated trajectory.



(a) Inevitable Collision State vs. Collision Free State

(b) ICS computation

Figure 7 Inevitable Collision States (ICS).

3. In case the newly obtained state s_n is safe, this control is valid.

The operation is repeated over all control inputs. Finally the new safe state s_n closest to the goal, is finally selected and added to the tree.

One difficulty when performing motion planning using an incremental approach is the choice of the metric used to find the closest states and expand the nodes when building the tree. This parameter is recognized to have a large influence on the trajectory quality specially when dealing with non-holonomic systems. Indeed, a typical euclidean metric does not account for the heading of the car. In case a L_∞based metric is used, the heading is weighted but does not account for a real non-holonomic displacement. Such a metric can yield to oscillatory and poor quality trajectories with low convergence (Fig. 6(b)). The first non-holonomic metric was based on real path length calculation, a summation of straight segments and arc of circles [4]. Later, the continuous curvature (CC) metric was presented in [25]. Basically, this metric connects the straight path and the arc of circle of the dubbins path, where there is a discontinuity in the curvature, with a clothoid. This metric is very natural to the system considered in our work and provides very high quality results (Fig. 6(a)).

3.5 Safety Issues

Like every method that computes partial motions, 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

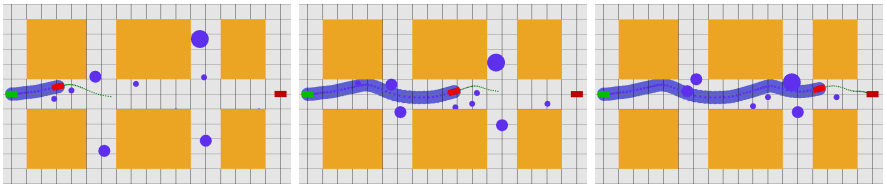


Figure 8 Navigation within an urban environment cluttered with moving pedestrians.

consider. In figure 7(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 [5], 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 [5]. 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. 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. 7(b)).

A safe trajectory therefore consists of a sequence of safe states, *ie.* ICS-free 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 [20] 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 an inevitable collision detection module within the exploration algorithms in place of the conventional geometric collision detection module. In (Fig.8), simulation results of partial safe trajectories of a car moving within a pedestrian environment are shown.

4 System Integration

The algorithms presented in §2 and §3, are implemented in C++ and integrated into a single application controlling an automated electric vehicle, the Cycab (Fig. 3(b)). The complete software runs on a standard 3.3 [GHz] PC.

Currently the only input data used is one layer of an IbeoML laser scanner. The distribution $P(S^x|\ o_t)$ is defined according to the manufacturer specifications. Measures are done at 8 [Hz] and the world model is updated in realtime. The trajectory update rate is set at 2 [Hz].

The tracking of the generated trajectories is ensured by a non-linear proportional closed loop controller (equation 7) . The control take advantage of the fact the

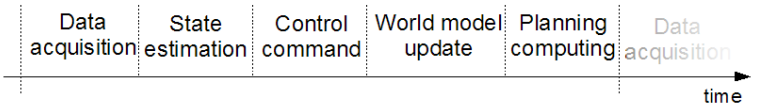


Figure 9 Execution time-line between two measurements.

trajectory planner computes a reference state including the desired speed value.

$$\begin{aligned} v &= v_{ref} \cdot (1 - k_1 \cdot \Delta_x) \\ \phi &= k_2 \cdot \Delta_y + k_3 \cdot \sin \Delta_\theta \end{aligned} \tag{7}$$

with v_{ref} the reference velocity given by the open loop trajectory calculated by PMP. The paramaters k_1, $k_2$and k_3 are tuning coefficients of the control law. The probable errors bounds of the tracking are integrated in the vehicle bounding box used during the planning stage in order to consider the collision risk associated to tracking errors.

Both perception and planning algorithms are designed to incrementally and iteratively construct a solution which enables an efficient and simple interweaving. Figure 9 present the task allocation sequence in the vehicle processor. As soon as a measure is available the vehicle state estimate is updated and the control command is sent. Computing the command signal as soon as possible allows to keep a good state estimate, and thus helping the control task. After the control stage, the world model is incrementally updated and processor time is allocated for the planning computation. During this time slot the planned trajectory is incrementally updated. Depending on the desired trajectory update rate a trade-off is made between reactivity and planning horizon. The execution cycle is executed under a soft real time constraint. 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 described perception algorithm is based on an optimization procedure. Fixing an maximum number of optimization steps allows to bound his computation time (currently set to 20 steps). In theory the method used to track moving obstacles grows cuadratically with the number of moving objects, however even for more than ten obstacles the computation time related to this task is neglible compared to the optimization step of the scan matching. The planning algorithm is also based on an optimization procedure which is interrupted on a periodic fashion. The output quality of both modules is proportional to the computation power (force in time) available. However as shown in the next section, current computational capabilities are enough for full scale experiments.

5 Experimental results

In figure 10, we present the result of an early experiment. The top of the 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 of the 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 vehicle is able to incrementally construct a map of approximately 50 square meters, define a safe partial trajectory towards the goal over some tens of meters and follow it, all in real time.

Observed execution times indicate that 40% of processing time is dedicated to perception, 40% to planning and 20% to graphical output (tracking cost is negligible). This gives average values of 50 [ms] for localization, mapping and moving objects tracking in the presented scenario and 200 [ms] to compute a trajectory for the next 500 [ms] avoiding the obstacles shown in the figure 10. Notice that in practice the trajectories found provides a solution for a time horizon much larger than 500 [ms] (around 20 times larger in the presented scenario).

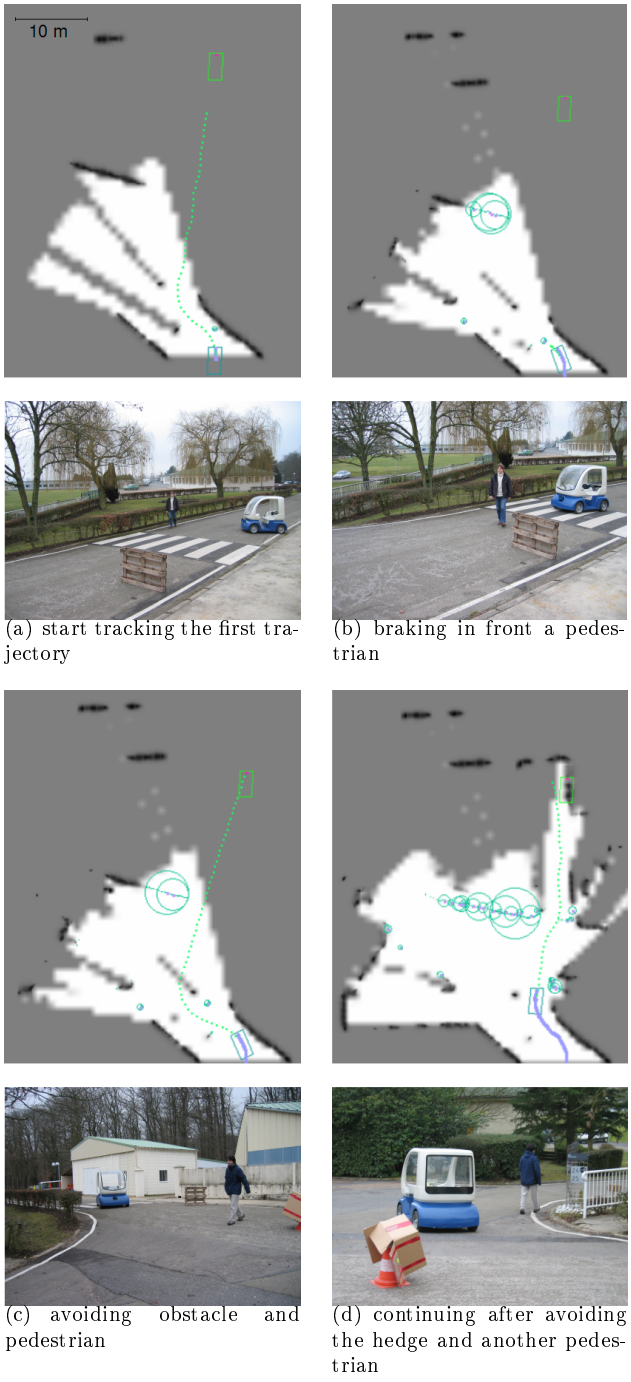
6 Toward city scale

We believe that the discussed perception and planning methods provide a sound solution to the problem of navigation in urban environments. However we have focused the discussion on the core issues related to small area navigation. One of the major remaining open task left is the extension of this work to the city scale level.

In the previous section it was shown that integrated perception and planning in medium sized urban areas is possible. The grow of computational power ensures that having better maps, trajectories and being able to deal with many moving obstacles is not an issue. As mentioned in section 2, in order to keep the local map error bounded a new map should be created periodically. Using previous measures to initialize the new map allows to have a continuous flow between successive local maps. This is only a technical issue related to the software implementation.

The perception and planning coupling presented here is a generic approach that can be applied to an environment of arbitrary complexity. For real world deployment ones has to ensure that the modeling of the moving objects and the inclusion of the non observed ones allows to do consistent predictions while having viable safe trajectories. If the vision range of the vehicles is too low, the vehicle will be not able to find a safe trajectory. The explicit elucidation of the interaction between safe planning, sensors precision and sensors range is left for a future document.

In order to reach city scale it is also necessary to connect the street level real time trajectory planning module to a city level road planner (such as commonly found in commercial navigation products). A non trivial issue arises with the fact that trajectory planning tries to reach a precise point on a local map, while current city level maps provide information within a few meters of error. Somehow the low precision path points have to be matched to the centimetric precision maps used for trajectory computation. Even if this is done, a second non trivial issue arises. We desire that the vehicle pass from one region to another as indicated by the higher level road planner. However, the proposed trajectory planning method



(a) start tracking the first trajectory

(b) braking in front a pedestrian

(c) avoiding obstacle and pedestrian

(d) continuing after avoiding the hedge and another pedestrian

Figure 10 Experiment results.

tries to reach a goal defined by a specific vehicle state (a specific point on the road). Extending the method to the notion of "goal region" is non trivial.

In order to solve these issues it is required to match the constructed high precision local map to the topologic description of the roads (used for road planning). While similar problems have been discussed in the context of aerial or satellital image processing [6, chapter 8] this specific issue remains an open problem. Even though the matching were realized the second issue would remain. A possible solution would consist in modifying the non-holonomic metric in order to integrate the notion of "goal region" instead of a specific goal state. This idea would provide a sound solution, however it seems complex and computationally more expensive. A second approach consists in trying to obtain, a priori, the number of lanes on the road. Then non-holonomic metric used in the trajectory planning can be modified to choose between a limited set of goals, one per lane center. This second idea, provides an easier but more application specific solution.

Matching between the topological description of the road and the constructed local maps can be done online. However there is a clear interest in constructing offline a city scale high resolution map. Such map would be used to enhance the trajectory planning since it would provide information about not yet observed regions. This map would also allow to obtain the number of lanes information. Furthermore, the matching to the topological roads description could be done offline, thus more robustly.

Most of the work on automatic large map construction has been centered on the SLAM problem. The objective is to compute an optimal map that reduces the uncertainty of position for each map portion. Much has been discussed around the computational cost associated to computing such maps and realizing online SLAM. However it is important to remind that the key issue for successful map construction is the data association problem. Optimizing online constructed map (i.e. doing online SLAM), reduces the pose uncertainty and thus helps to do data association. If data association was always successful offline map optimization would suffice. Existing commercial GPS systems allow to bound the uncertainty of the position anywhere over the globe to less than 30 meters of error. However these systems require a wireless link with satellites, which are commonly unreachable in dense cities [3]. A realistic estimation is 50% of availability. Successful city sized SLAM will depend on the adequate fusing of GPS measures with other sensors measures in order to bound the pose uncertainty up to the attraction region of the data association algorithm. An interesting research direction consists in enhancing the data association method to be able to deal with GPS grade position errors. Since the non availability regions of the GPS are variable, 100% success rates will be not possible to ensure. In such situations erroneous data association detection and correction methods can be applied [10].

7 Conclusion and perspectives

In this paper, we analyzed the main difficulties associated to the navigation of driverless vehicles in urban environments. We 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 an experimental validation of the proposal.

As discussed through this text we identify four areas for future work: constructing high precision city scale maps at low cost; coupling road and trajectory planning; fusing multiple sensors for enhanced localization; fusing multiple sensors for enhanced road, sidewalks and obstacles detection.

We believe that working on these key areas will open the door for city scale driverless vehicles.

References

[1] 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.

[2] 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.

[3] Jason Chao, Yong qi Chen, Wu Chen, Xiaoli Ding, Zhilin Li, Nganying Wong, and Meng Yu. An experimental investigation into the performance of gps-based vehicle positioning in very dense urban areas. *Journal of Geospatial Engineering*, 3:59–66, 2001.

[4] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. 79:497–517, 1957.

[5] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[6] Christian Frueh. *Automated 3D Model Generation for Urban Environments.* PhD thesis, University of Karlsruhe, 2002.

[7] 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.

[8] 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.

[9] Dirk Haehnel. *Mapping with Mobile Robots.* PhD thesis, Fakultat fuer Angewandte Wissenschaften, Universitat Freiburg, December 2004.

[10] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy, 2003. Springer.

[11] Arie Kaufman Haitao Zhang, Olaf Hall-Holt. Range image registration via probability field. In *Computer Graphics International Conference*, pages 546 – 552, 2004.

[12] 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.

[13] S. Kolski, D. Ferguson, M. Bellino, and R. Seigwart. Autonomous driving in structured and unstructured environments. 2006. to appear.

[14] 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.

[15] 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.

[16] J. Minguez, F. Lamiraux, and L. Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *Int. Conf. on Robotics and Automation*, Barcelona, Spain, 2005.

[17] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, San Jose, CA, 2004. AAAI.

[18] L. Montesano, J. Minguez, 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.

[19] M. Parent. Automated public vehicles : A first step towards the automated highway. In *4th World Congress on Intelligent Transport Systems*, October 1997.

[20] 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.

[21] Roland Philippsen, Björn Jensen, and Roland Siegwart. Toward online probabilistic path replanning in dynamic environments. In *IROS Proceedings*, 2006.

[22] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier. The cycab: a car-like robot navigating autonomously and safely among pedestrians. *Robotics and Autonomous Systems*, 50(1):51–68, 2005.

[23] 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.

[24] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Int. Conf. on 3D Digital Imaging and Modeling*, 2001.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] S. Thrun, M. Bennewitz, W. Burgard, A.B. 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 *Int. Conf. on Robotics and Automation (ICRA)*, 1999.

[30] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, 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. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. accepted for publication.

[31] 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.

[32] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[33] Xiaogang Wang, Kinh Tieu, and Eric Grimson. Learning semantic scene models by trajectory analysis. Technical report, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2006.

[34] Denis F. Wolf and Gaurav S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, July 2005.

[35] Zhengyou Zhang. Iterative point matching for registration of free-form curves. Technical Report RR-1658, INRIA-Sophia Antipolis, April 1992. Equipe : ROBOTVIS.