Modeling Dynamic Scenarios for Local Sensor-Based Motion Planning

Luis Montesano[a] Javier Minguez[b] Luis Montano[b]

[a]*Instituto de Sistemas e Robotica, Instituto Superior Tecnico, Lisboa, Portugal*

[b]*Dpto de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain*

Abstract

This paper addresses the modeling of the static and dynamic parts of the scenario and how to use this information with a sensor-based motion planning system. The contribution in the modeling aspect is a formulation of the detection and tracking of mobile objects and the mapping of the static structure in such a way that the nature (static/dynamic) of the observations is included in the estimation process. The algorithm provides a set of filters tracking the moving objects and a local map of the static structure constructed on line. In addition, this paper discusses how this modeling module is integrated in a real sensor-based motion planning system taking advantage selectively of the dynamic and static information. The experimental results confirm that the complete navigation system is able to move a vehicle in unknown and dynamic scenarios. Furthermore, the system overcomes many of the limitations of previous systems associated to the ability to distinguish the nature of the parts of the scenario.

Key words: Mobile robots, mapping dynamic environments, sensor-based motion planning

1 Introduction

Autonomous robots are currently being deployed in real environments such as hospitals, schools or museums developing help-care or tour-guide applications for example. A common characteristic of these applications is the presence of people or other moving objects. These entities make the environment dynamic and unpredictable and have an impact on the performance of many of the basic robotic tasks. One of these tasks, common for many robotic applications, is mobility. In particular, dynamic scenarios involve two aspects in the motion generation context: (*i*) to model the scenario and (*ii*) to integrate this information within the motion layer. This paper addresses both issues.

The majority of existing motion systems address dynamic scenarios by using the sensor observations at high rate compared to the obstacles dynamics. In other words, they assume a static but rapidly sensed scenario, which allows fast reactions to the changes induced by the evolution of the moving objects. Although this assumption works fine in many cases (obstacles moving at a low speed), in realistic applications is no longer valid. This is because in reality the object's motion is arbitrary and, even assuming low speeds, in many cases these systems fail. For instance, Figure 1 shows two common and simple situations where the static and dynamic parts of the scenario have to be discriminated, modeled and consequently used within the motion generation layer. To explicitly deal with dynamic objects is a must to improve the robustness of the motion systems.

In this work we are interested in those applications where the scenarios are dynamic and unpredictable and, thus, require rapid reactions of the vehicles. For instance, consider a robotic wheelchair (Figure 2). In this type of application, the user places goal locations that the wheelchair autonomously attains. In this context, the goals are usually in the field of view of the user, that is, in the close vicinity of the robot. This is important to bound the spatial domain of the motion generation. In general, the motion task has been usually addressed from a global or local point of view (i.e. global or local motion systems). For full autonomous operation both are required since they are complementary, however their competences are different and related with the spatial domain and the reaction time. In short, the larger the spatial domain is, the higher the reaction time due to the computational requirements [1].

On the one hand, global systems address solutions with large spatial domains. In static environments, successful global mapping and planning has been demonstrated even though the computational requirements increase with the size of the scenario. Dynamic scenarios impose additional difficulties since it has been proved that the motion planning problem in the presence of dynamic obstacles is NP-hard [10] (even in simple cases such as a point robot

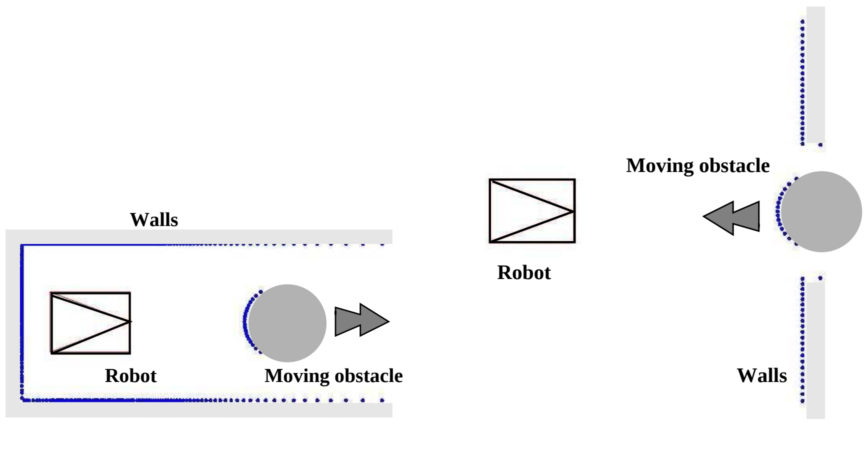


Fig. 1. These figures depict two examples that illustrate the importance of modeling and using the dynamic and static parts of the scenario in the motion layer. The points are laser measurements. Situations where (a) the robot and a dynamic obstacle move in a long corridor, and (b) the robot moves toward a door that is temporally blocked by a moving obstacle. Without distinction between static and dynamic obstacles, both the corridor and the door seem to be blocked and every motion layer using the sensor measurements without processing would fail. To solve both situations, we need to construct a model of the static and dynamic parts of the scenario and use them consequently in the motion layer (adapting the motion to the object dynamics).

and convex polygonal moving obstacles). In other words, global mapping and planning are time consuming operations especially in dynamic scenarios. Thus, they are not adapted to high rate perception - action operations.

On the other hand, in local systems the domain is usually bounded to achieve high rate perception-action schemes (working within the control loop). Furthermore, the motion problem in dynamic and unpredictable scenarios is local in nature, since: (*i*) it makes no sense to maintain a map of dynamic objects observed long time ago (e.g. two rooms away from the current location); and (*ii*) moving obstacles can modify the route arbitrarily (e.g. humans) constantly invalidating plans. That is why we focus our attention on local motion systems. These systems work within a high frequency perception - action scheme. Real-time is achieved by limiting the model size (used to plan the motions) and alleviating some constraints of the planner (to speed it up). The consequences of these design choices are: (*i*) the maximum reach of the motion solution is the model size, and (*ii*) the system might fail in some rare situations due to the under-constrained motion planning strategies. Despite these limitations, local systems are able to compute robust local motion in the large majority of cases (see [43] for a discussion). Additionally, as discussed before, these local techniques can be combined with global techniques improving the behavior in long-term missions. In particular, in the robotic wheelchair application, the

[redacted]

Fig. 2. A child using an autonomous robotic wheelchair in a crowded and populated scenario. The number and dynamics of the moving obstacles is unpredictable. To deal with the motion aspect in these scenarios is the context of this work.

high level (global) competences may rely on a human and the machine addresses local issues. This paper focuses on the construction of local models in dynamic environments and their integration with local motion planning systems.

The paper is organized as follows. Section 2 describes the related work and the contributions of this work. Section 3 presents the modeling of the static and dynamic scenarios, and in Section 4 this module is integrated with a local motion planning system. In Section 5 we describe the experimental results to validate the modeling and the motion generation in dynamic scenarios. Section 6 draws the conclusions.

2 Related Work and Contributions

This paper focuses on two aspects of the design of motion systems in dynamic scenarios: (*i*) to appropriately model the scenario, and (*ii*) to consequently integrate and use this model in the motion layer. Thus, we articulate this section in these two directions and, finally, we outline the contributions of this work.

2.1 Modeling Dynamic Scenarios

Modeling dynamic scenarios has at least two aspects: the modeling of the static parts of the environment and the identification and tracking of the moving objects. On one hand, the modeling of static scenarios has been extensively studied. The proposed algorithms include incremental maximum likelihood mapping techniques [20,19,25], batch mapping algorithms [22] or online Simultaneous Localization and Map Building (SLAM) [12,11,49]. On the other hand, the Tracking of Moving Objects (TMO) is also a well-studied problem [3,8]. However, a robust modeling of dynamic environments requires to perform both tasks at the same time. This is because the robot position error affects the classification of the measurements and, consequently, the map construction and the tracking of the moving objects.

The modeling techniques that address both issues simultaneously can be roughly divided into global or local techniques. On one hand, there are global techniques that address the mapping and tracking problem simultaneously. For example, [52] presents a rigorous formulation of the TMO-SLAM problem assuming a known classification of the observations into static and dynamic. The problem is factorized into an independent SLAM and an independent tracking. In the implementation, the classification is based on the violation of the free space in a local dense grid map and the tracking on a set of extended Kalman filters (EKF). In [21] they use a feature based approach to detect the moving objects in the range profile of the laser scans. Next, they use Joint Probabilistic Data Association particle filters [44] to track the moving objects and a probabilistic SLAM technique to build the map. The previous methods do not take into account the uncertainty of the robot motion in the classification step. Thus, difficulties may arise in the presence of large odometry errors due to misclassification, since these errors affect the precision and the convergence of the previous algorithms. Incorporating the classification process within the estimation process is hard due to the combination of discrete and continuous hidden variables and results in intensive computing algorithms. For instance, in [22], an Expectation Maximization based algorithm filters the dynamic measurements that do not match the current model of the static parts while building a map of the environment. However, this technique is not well suited for real time motion generation because of its batch nature and because it does not explicitly model the dynamic features.

On the other hand, the usual strategy with the local techniques is to build the map by filtering the measurements originated by the moving objects. Many algorithms use a scan matching technique to correct the robot position and incrementally build a local map [5,26,7,32]. The performance of these techniques is affected by dynamic environments due to failures in the matching process. Several authors have extended them to minimize the effect of the moving objects by discarding sectors with big correspondence errors [4] or by using the Expectation Maximization (EM) method to detect and filter the spurious measurements [23]. These type of methods are widely spread and used, since they are well adapted to real time operation. However, they discard dynamic information and do not explicitly track the moving objects. The lack of a model for the dynamic parts hampers their filtering in subsequent measurements and prevents the usage of this information for other tasks (e.g. sensor-based motion planning).

2.2 Motion Generation in Dynamic Scenarios

The correct way to generate motion in dynamic scenarios is to address the motion planning with moving obstacles. Unfortunately it has been demonstrated that this problem is NP-hard even for the most simple cases (point robot and convex polygonal moving obstacles [10]). As a result, these techniques are not well suited for real time operation. This is because they take significant time to compute the plan and often, when it is available, the scenario has been modified invalidating the plan. The problem of motion in dynamic scenarios is usually simplified to achieve real time operation.

The usual simplification is to compute the motion with reactive obstacle avoidance techniques, which reduce the complexity of the problem by computing only the next motion (instead of a full plan). Therefore, they are very efficient for real-time applications. Some reactive techniques have been designed to deal with moving obstacles [16,18] and have demonstrated good performance. Unfortunately their local nature produces trap situations and cyclic behaviors, which is a strong limitation for realistic operation.

To overcome this limitation it has been suggested to combine these reactive techniques with some sort of planning (see [1] for a discussion on integration schemes and [30] for a similar discussion in the motion context). The more widespread way to combine reaction with planning are the systems of tactical planning [42,50,9,30,46,41,37]. They perform a rough planning over a local model of the scenario, which is used to guide the obstacle avoidance. The planning extracts the connectivity of the space to avoid the trap situations. The reactive collision avoidance computes the motion addressing the vehicle constraints. The advantage of these local motion systems is that the combination of planning and obstacle avoidance at a high rate assures robust collision avoidance while being free of local minima (up to some rare cases related with the relaxation of constraints of the planning algorithm [43]). However, none of these systems constructs models of the dynamic and static parts of the scenario. Thus, they cannot correctly cope with some typical situations (such as those depicted in Figure 1) affecting the robustness of the system.

2.3 Contributions

This paper contributes in two aspects of the motion generation where the dynamic obstacles affect: (*i*) the construction of a model of the dynamic and static parts of the scenario and (*ii*) its integration within a local system of tactical planning.

- The first contribution is an incremental local mapping algorithm that explicitly solves the classification process. The method uses the Expectation Maximization algorithm to compute the robot pose and the measurement classification, constructs a local dense grid map and tracks the moving objects around the robot. The method could also be seen as a scan matching algorithm for dynamic environments that includes the information about the moving objects.
- The second one is the integration of the previous modeling within a local sensor-based motion system based on a tactical planning scheme. The advantage is that the motion is computed by selectively using the information provided by the static and moving obstacles in the planning - obstacle avoidance paradigm. As a consequence, the system is able to drive the vehicle in dynamic scenarios while avoiding typical shortcomings classically related with mobility such as the trap situations or the motion in confined spaces.

3 Modeling dynamic environments

In this section, we outline the problem of modeling dynamic environments from a Bayesian perspective. Next, we present a maximum likelihood algorithm to jointly estimate the robot pose and classify the measurements into static and dynamic; and we provide the implementation details for a laser sensor.

The objective is to estimate the map of the static parts and the map of the moving objects around the robot using the information provided by the onboard sensors. Formally, let $Z_k = \{z_{k,1}, ..., z_{k,N_z}\}$ be the N_z observations obtained by the robot at time k and u_k the motion command executed at time k. The sets $Z_{1:k} = \{Z_1, ..., Z_k\}$ and $u_{0:k} = \{u_0, ..., u_k\}$ represent the observations and motion commands up to time k. Let x_k denote the robot location at time k, $O_k = \{o_{k,1}, ...o_{k,N_O}\}$ the state of the N_O moving objects at time k and M the map of the static environment[1] . From a Bayesian point of view,

[1] The assumption here is that the map M does not change over time (note that the map does not have a time index). The formulation states that the world can be divided in two different types of features: static and dynamic. The static ones are parameters (their value is fixed) while the dynamic ones require modeling their evolution. This assumption is common in this context and in most of the algorithms

the objective is to estimate the distribution $p(O_k, x_k, M \mid Z_{1:k}, u_{0:k-1})$. Using the Bayes rule and marginalizing out the state variables at the previous step, we get the recursive Bayes estimator

$$\begin{aligned} p(O_k, x_k, M \mid Z_{1:k}, u_{0:k-1}) &= \eta p(Z_k \mid O_k, x_k, M) \qquad (1) \\ \int\int p(O_k, x_k \mid O_{k-1}, x_{k-1}, M, u_{k-1}) & p(O_{k-1}, x_{k-1}, M \mid Z_{1:k-1}, u_{0:k-2}) dO_{k-1} dx_{k-1} \end{aligned}$$

where η is a normalization factor. The term $p(Z_k \mid O_k, x_k, M)$ is known as the measurement model. The integral combines the motion model $p(O_k, x_k \mid O_{k-1}, x_{k-1}, M, u_{k-1})$ and the distribution of interest at $k-1$ to predict the state vector at time k. Notice that, since the map does not change over time, it is a constant in the integration. We have used a Markov assumption to discard previous measurements and commands and simplify both the motion and measurement models.

The motion model, $p(O_k, x_k \mid O_{k-1}, x_{k-1}, M, u_{k-1})$, represents the evolution of the robot and the moving objects. Let us assume that the objects $o_{k,i}$ and the robot x_k move independently. Then, the joint motion model can be factorized into the individual motion models of the robot and each moving object. If the motion does not depend either on the map M, the motion model can be written as

$$p(O_k, x_k \mid O_{k-1}, x_{k-1}, u_{k-1}) = p(x_k \mid x_{k-1}, u_{k-1}) \prod_{i}^{N_z} p(o_{k,i} \mid o_{k-1,i}) \qquad (2)$$

The likelihood term, $p(Z_k \mid O_k, x_k, M)$, measures how well the observations match the prediction done by the motion model. Computing this term requires to solve the data association problem, i.e. to establish a correspondence between each measurement and a feature of the map or a moving object. In order to model the data association, we introduce a new variable c_k that indicates which feature originated each measurement Z_k. Since the correspondences are unobserved, one has to integrate over all the possible sources c_k

$$p(Z_k \mid O_k, x_k, M) = \sum_{c_k} p(Z_k, c_k \mid O_k, x_k, M) p(c_k \mid O_k, x_k, M) \qquad (3)$$

that map static environments. Otherwise, if all features are considered dynamic, the non-visible parts of the map become unusable after some time since their location tends to a non informative distribution (their uncertainty increases in an unbounded way). Furthermore, observability also becomes an issue due to the uncertainty in the robot displacement.

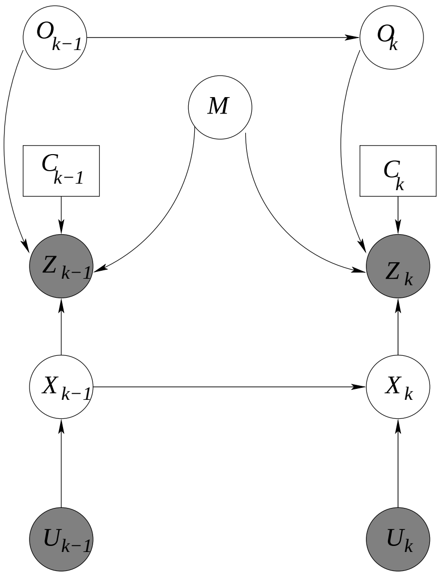


Fig. 3. Graphical representation of the problem including the data associations. The circles represent the continuous variables and the squares discrete ones. The filled nodes are the measurements whereas the empty ones represent hidden variables.

Figure 3 shows the graphical representation of the problem. The model contains the continuous variables of Equation (1) and the discrete variables c_k representing the source of the measurements at each point in time (i.e. they represent the classification or correspondence problem into static/dynamic).

In general, it is very hard to perform exact inference on such models due to the integration over all the possible correspondences and the mutual exclusion constraints. Therefore, one has to use approximations. It is possible to use sequential Monte Carlo techniques [14,39] to approximate the full distribution. The drawback is a high computational cost due to the increasing size of the map and the multiple hypotheses arising from different classifications. As described in Section 2, most of previous works simplify the problem assuming that the classification into static and dynamic is known. In practice the classification is not available and it is usually computed based on the distribution $p(O_k, x_k, M \mid Z_{1:k-1}, u_{0:k-1})$ using patterns and/or free space constraints.

In the next section, we propose an incremental mapping algorithm that jointly computes the robot pose x_k and the correspondences c_k to obtain the estimate of the map M and the location of the moving objects O_k.

3.1 Incremental mapping of dynamic environments

A key feature of previous approaches is a high computational time due to the increasing size of the map, which hinders its application in the context of this work (real - time operation). One simplification of the full modeling problem is incremental mapping [48,19]. The objective is to compute a single pose $\hat{x}_k$ of the vehicle at each point in time k (instead of a distribution). Thus, the representation of the map and the moving objects, which can be probabilistic, are conditioned on this deterministic trajectory.

Incremental mapping estimates at each point in time k the robot pose $\hat{x}_k$ that maximizes the likelihood term $p(Z_k \mid O_k, x_k, M)$

$$\hat{x}_k = arg \max_x p(Z_k \mid O_k, x_k, M) p(x_k \mid \hat{x}_{k-1}, u_{k-1}) \tag{4}$$

The additional term $p(x_k \mid \hat{x}_{k-1}, u_{k-1})$ introduces the uncertainty of the last robot motion u_{k-1} to constraint the possible solutions of the optimization algorithm. Furthermore, the map M and the state of the objects O_k are computed from the set of measurements $Z_{1:k-1}$ and poses $\hat{x}_{1:k-1}$ up to time $k-1$,

$$M_k = f_M(\hat{x}_{1:k-1}, Z_{1:k-1}) \tag{5}$$
$$O_k = f_O(\hat{x}_{1:k-1}, Z_{1:k-1}) \tag{6}$$

The robot trajectory $\hat{x}_{1:k-1}$ used by functions f_M and f_O is the deterministic set of maximum likely poses. The detailed description of functions f_M and f_O is given in the next sections. However, let us advance that for computational reasons, they are incremental functions that use the estimates at $k-1$, the new pose and the last set of measurements.

The advantage of this framework is that the classification of the measurements can be included within the maximum likelihood optimization using the Expectation-Maximization (EM) algorithm. This is addressed in the next section.

3.2 Expectation Maximization (EM) Maximum Likelihood approach

So as to solve Equation (4) the term $p(Z_k \mid O_k, x_k, M)$ has to be maximized, which requires to consider all the possible sources (static map or a dynamic object) for each observation (see Equation (3)). The resulting expression has no closed-form solution and it is difficult to maximize since the classification variables c_k are unknown in general (Figure 3). This subsection describes how to use the Expectation Maximization (EM) [13,27] formulation to address the static/dynamic classification process and solve Equation (4) to obtain $\hat{x}_k$.

The EM technique is a maximum likelihood approach to solve incomplete-data optimization problems. Initially, it introduces some auxiliary variables to convert the incomplete-data likelihood into a complete-data likelihood L_c which is easier to optimize. Then, there is a two step maximization process: (i)

the E-step computes the conditional expectation $Q(x, x^{(t)}) = E_{x^{(t)}}[\log L_c \mid Z_k]$ of the complete-data log-likelihood given the measurements Z_k and the current estimate $x^{(t)}$ of vector x to be maximized; and (ii) the M-step computes a new estimate $x^{(t+1)}$ that maximizes the function $Q(x, x^{(t)})$. This process is repeated until the change in likelihood in the complete-data likelihood is arbitrarily small. The original incomplete likelihood is assured not to decrease after each iteration and, under fairly regular assumptions the algorithm converges to a local maximum [27].

In the remainder of this section, we describe the application of the EM technique to maximize the likelihood term $p(Z_k \mid O_k, x_k, M)$ of Equation (4). The second term, $p(x_k \mid \hat{x}_{k-1}, u_{k-1})$, is just a prior over the robot poses. Since it does not depend on the measurements, it only affects the M-step (Section 3.3.3) acting as a regularization term [2].

Initially, we define some extra variables to build the complete-data likelihood function. Although, in general the extra variables in the EM does not require to have any physical meaning, in our case we use the correspondence variable c_k. This is because if this variable is known, the resulting likelihood is much simpler and its maximization has a closed form solution (given that we linearized the measurement equation).

Let the correspondence variable c_k be a vector of binary variables c_{ij} with $j \in 0..N_o + 1$ defined for each observation $z_{k,i}$, $i \in 1..N_z$ of Z_k. So as to ease the notation, we drop the time index k from the binary variables c_{ij}. The possible sources are represented by the index j, where $j = 0$ for static measurements, $j \in 1..N_O$ for the tracked objects, $j = N_O + 1$ for the unknown sources. The value $c_{ij} = 1$ indicates that $z_{k,i}$ was originated by source j, and $c_{ij} = 0$ otherwise.

Assuming that a single observation only belongs to one source (i.e. $\sum_j c_{ij} = 1$) and that the measurements $z_{k,i}$ are independent, the complete-data likelihood model is

$$L_c = p(Z_k, c_k \mid x_k, M_k, O_k) = \prod_{i=1}^{M} \left[p_s(z_{k,i} \mid x_k, M_k)^{c_{i0}} p_u(z_{k,i})^{c_{iN_O+1}} \prod_{j=1}^{N} p_d(z_{k,i} \mid x_k, o_{k,j})^{c_{ij}} \right] \tag{7}$$

where $p_s(.)$, $p_d(.)$ and $p_u(.)$ are the likelihood models for observations originated from the map (static), each of the tracked moving objects (dynamic) and new discovered areas (unknown), respectively. Note that the binary variables

[2] Under Gaussian assumptions, the inclusion of this term in the minimization still has a closed form solution.

c_{ij} select the likelihood model according to the origin of the measurements. The complete-data log likelihood function is

$$\log L_c = \sum_{i=1}^{N_z} \Bigg[c_{i0} \log p_s(z_{k,i} \mid x_k, M_k) + c_{iN_O+1} \log p_u(z_{k,i}) + \sum_{j=1}^{N_O} c_{ij} \log p_d(z_{k,i} \mid x_k, o_{k,j}) \Bigg] \quad (8)$$

The function $Q(x_k, x_k^{(t)})$ is the conditional expectation of the complete-data log likelihood $\log L_c$. As $\log L_c$ is linear in the unobservable data c_k, in the computation of $Q(x_k, x_k^{(t)})$ we replace each c_{ij} by its conditional expectation given the current measurements Z_k and the current estimates of x_k, M_k and O_k,

$$\begin{aligned} Q(x_k, x_k^{(t)}) &= E_{x_k^{(t)}}\{logL_c \mid Z_k, M_k, O_k\} \quad (9) \\ &= \sum_{i=1}^{N_z} \Bigg[\hat{c}_{i0} \log p_s(z_{k,i} \mid x_k, M_k) + \hat{c}_{iN_O+1} \log p_u(z_{k,i}) + \sum_{j=1}^{N_O} \hat{c}_{ij} p_d(z_{k,i} \mid x_k, o_{k,j}) \Bigg] \end{aligned}$$

where

$$\hat{c}_{i0} = E_{x_k^{(t)}}\{c_{i0} \mid Z_k, M_k, O_k\} \quad (10)$$
$$\hat{c}_{iN_O+1} = E_{x_k^{(t)}}\{c_{iN_O+1} \mid Z_k, M_k, O_k\} \quad (11)$$
$$\hat{c}_{ij} = E_{x_k^{(t)}}\{c_{ij} \mid Z_k, M_k, O_k\} \quad (12)$$

3.3 Implementation for range sensors

In this Section we provide the implementation of the previous method for a range sensor, e.g. laser range finder. First, we present the measurement models associated to each type of measurement (static, dynamic, unknown). Based on these models, we derive the equations for the E-Step and M-Step of the EM algorithm based on the function $Q(x_k, x_k^{(t)})$ of the previous section. Finally, we describe how to update the map M_k and the moving objets O_k using the new pose $\hat{x}_k$. Algorithm 1 summarizes the steps of the algorithm.

3.3.1 Models

We next address the implementation of the functions $f_M(\cdot)$ and $f_O(\cdot)$ (see Equation (5)), which compute the current estimates of the map M_k and the

Algorithm 1 : Algorithm Summary

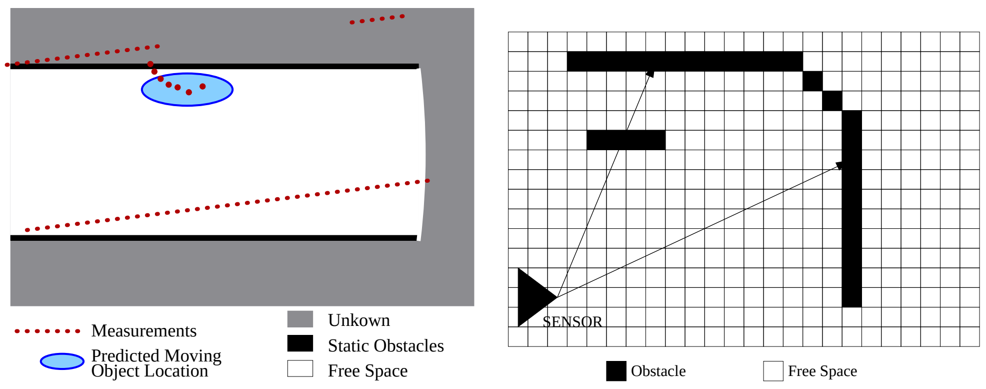
```
INPUT: $x_{k-1}$, $u_{k-1}$, $Z_k$, $M_{k-1}$, $O_{k-1}$
$t = 0$,
% Prediction step
Compute the initial $x_k^{(0)}$ using $x_{k-1}$ and $u_{k-1}$
Predict moving objects locations $O_{k|k-1}$
% Estimation of the robot pose
repeat
  E-Step:
  for each $z_{i,k}$, do
      Select the nearest occupied grid cell $g_i \in M_{k-1}$ using the Mahalanobis
      distance
      Compute the Mahalanobis distance to each $o_j \in O_{k|k-1}$, $j \in 1..N_O$
      Compute $\hat{c}_{i0}, \hat{c}_{ij}, \hat{c}_{iN_{O+1}}$ to form $Q(x_k, x_k^{(t)})$
  end for
  M-Step:
  Compute $x^{(t+1)} = \arg max_{x_k} Q(x_k, x_k^{(t)})$
  $t = t + 1$
until convergence or $t > MaxIter$
% Update models using $x_k = x^{(t)}$
Classify measurements into $Z^{static}, Z^{dynamic}$
Update $M_k$ with static measurements $Z^{static}$
Update filters $O_k$ with dynamic measurements $Z^{dynamic}$
OUTPUT: $x_k$, $M_k$, $O_k$
```

moving objects O_k conditioned over the set of robot poses $\hat{x}_{1:k-1}$. Next, we will describe the likelihood terms $p_s(.)$, $p_d(.)$ and $p_u(.)$ of Equation (7).

For the static map $f_M(\cdot)$, we use a two dimensional probabilistic grid [38] to represent the workspace. Each cell has associated a random binary variable m_i, where $m_i = 1$ when it is occupied by a static obstacle, and $m_i = 0$ if it is free space. The probability of the grid cells is computed using the Bayesian approach proposed in [15]. This map representation is convenient in our navigation context since: (i) it contains information of occupied and free space, and (ii) it is suitable for unstructured scenarios.

We implement the function $f_O(\cdot)$ using an independent extended Kalman filter to track each of the moving objects. The state vector for each moving object contains its position, its velocity and its size. The function $f_O(\cdot)$ computes the predicted positions of the moving objects at time k based on the vehicle trajectory $\hat{x}_{1:k}$ and the measurements $Z_{1:k-1}$. We use a constant velocity model with acceleration noise to predict the positions of the moving objects between observations and a random walk model for the size of the object.

The complete-data likelihood function of Equation (7) represents how well the observations fit the current estimate of the environment. This expression



(a) (b)

Fig. 4. (a) This figure illustrates how the error in the robot pose hinders the classification of the measurements. In particular, the effect of rotation error increases for points far away from the sensor. The figure also shows how the information of the object predicted position helps to improve the correspondences. (b) This figure illustrates the differences between an end-point model and a complete one. In the case of an end point model, both beams have the same probability. On the other hand, a complete model will assign a lower likelihood to the top ray due to the fact that it traverses an obstacle before reaching the final obstacle.

explicitly reflects the different possible sources for each measurement with the models $p_s(.)$, $p_d(.)$ and $p_u(.)$ and the classification variables c_k. The definition of the measurement models depends on the previous representations and on the sensor used (for instance, see [22,49] for laser range sensors). We use a correspondence oriented model where each likelihood term of Equation (7) is computed based on the Mahalanobis distance between each observation and its model conditioned on c_k. We model the uncertainties using Gaussian distributions and linearize the models to compute the Mahalanobis distance as in [36] (see Appendix A).

This framework is convenient since: (i) the Mahalanobis distance takes into account the measurement noise and the error of the last robot displacement, which may have a big impact on the classification of each measurement (Figure 4a); and (ii) the use of a probabilistic metric improves the correspondences computed in the E-step resulting in a better convergence and robustness [36].

In the previous model, we implicitly make two simplifications to decrease the computational requirements of the algorithm: (i) we use an *End Point model* [49], which ignores the path traversed by the ray and only considers the end point (Figure 4b); (ii) instead of taking into account all the possible correspondences between the measurements and the map of static obstacles in Equation (7), we select the nearest neighbour occupied cell of the map for each measurement. Although this is a simplification, nothing prohibits the usage of more sophisticated techniques such as [23,35] in the framework.

We next describe the models for each type of measurements $p_s(.)$, $p_d(.)$ and $p_u(.)$. The likelihood of a measurement associated to a cell of the static map is modeled as a Gaussian,

$$p_s(z_{k,i} \mid x_k, M_k) = N(z_{k,i}; f(x_k, q_i), P_{i0}) \tag{13}$$

where q_i is the location of the correspondent point associated to the measurement $z_{k,i}$, $f(x_k, q_i)$ is the transformation between the map and the robot reference systems. The covariance matrix P_{i0} takes into account the uncertainty in the last robot motion, the position of the point in the map and the measurement noise. The computation of P_{i0} is described in Appendix A. We refer the reader to [36] for further details.

We use the same model as the likelihood function for every dynamic object,

$$p_d(z_{k,i} \mid x_k, o_{k,j}) = N(z_{k,i}; f(x_k, o_{k,j}), P_{ij}), \quad \forall j \in 1..N_O \tag{14}$$

where the function $f(\cdot,\cdot)$ and the covariance matrix P_{ij} are the same as in Equation (13) but applied to the estimated position of the moving object $o_{k,j}$ (see Appendix A).

Finally, the classification variable $c_{iN_{O+1}}$ includes spurious observations and those corresponding to unexplored areas and new moving objects. The likelihood of such a measurement is difficult to quantify. It depends on the spurious rate of the sensor and on where the measurement is,

$$p_u(z_{k,i}) = \begin{cases} p_{unexplored} & \text{if } z_{k,i} \notin M_k \\ p_{spurious} & \text{if } z_{k,i} \in M_k \end{cases} \tag{15}$$

The value $p_{spurious}$ is an experimental value measuring the spurious rate of the sensor and $p_{unexplored}$ is typically set to a higher value to avoid those observations placed in unknown areas to influence the optimization process.

3.3.2 E-Step

The E-Step requires the computation of the expectation of the classifications variables $\hat{c}_{ij}$ defined in Equations (10-12),

$$\hat{c}_{ij} = E_{x_k^{(t)}}\{c_{ij} \mid Z_k, x_k, M_k, O_k\}$$

$$= \sum_{c_{ij}} c_{ij} p(c_{ij} \mid Z_k, x_k, M_k, O_k) = p(c_{ij} = 1 \mid Z_k, x_k, M_k, O_k) \tag{16}$$

where $i = 1..N_z$ and $j = 0..N_O + 1$. The expectation is conditioned on the predicted position of the objects O_k, the last map estimate M_k and the current estimate of x_k. Using the Bayes rule, we obtain

$$\begin{aligned} p(c_{ij} = 1 \mid Z_k, x_k, M_k, O_k) &= \frac{p(z_{k,i} \mid c_{ij} = 1, x_k, M_k, O_k) p(c_{ij} = 1 \mid x_k, M_k, O_k)}{p(z_{k,i} \mid x_k, M_k, O_k)} \\ &= \frac{p(z_{k,i} \mid c_{ij} = 1, x_k, M_k, O_k) p(c_{ij} = 1)}{\sum_l p(z_{k,i} \mid c_{il} = 1, x_k, M_k, O_k)} \end{aligned} \tag{17}$$

The previous derivation assumes a constant value for the prior over the classification variables $p(c_{ij} = 1 \mid x_k, M_k, O_k)$ and computes the probability of the observation, $p(z_{k,i} \mid x_k, M_k, O_k)$, as the sum of all its potential sources. The specific likelihood model to be used ($p_s(\cdot)$, $p_d(\cdot)$ or $p_u(\cdot)$ defined in Equations (13), (14) or (15)) depends on the source of the measurement indicated by the variable c_{ij}.

3.3.3 M-Step

The M-Step computes a new robot pose $x_k^{(t+1)}$ such that

$$Q(x_k^{(t+1)}, x_k^{(t)}) \geq Q(x_k, x_k^{(t)}) \tag{18}$$

Given the models introduced in Section 3.3 and Equation (9), the criterium to minimize is,

$$\begin{aligned} Q(x_k, x_k^{(t)}) = \sum_{i=1}^{N_z} \Bigg[& \hat{c}_{i0} \log \left(-2\pi \sqrt{|P_{i0}|} \right) \\ & + \hat{c}_{i0} (f(x_k, q_i) - z_{k,i})^T P_{i0}^{-1} (f(x_k, q_i) - z_{k,i}) + \hat{c}_{iN_O+1} \log p_u(z_{k,i}) \\ & + \sum_{j=1}^{N_O} \left[\hat{c}_{ij} \log \left(-2\pi \sqrt{|P_{ij}|} \right) + \hat{c}_{ij} (f(x_k, o_{k,j}) - z_{k,i})^T P_{ij}^{-1} (f(x_k, o_{k,j}) - z_{k,i}) \right] \Bigg] \end{aligned} \tag{19}$$

Grouping all the terms that do not depend on x_k we get

$$Q(x_k, x_k^{(t)}) = cte + \sum_{i=1}^{N_z} \Bigg[\hat{c}_{i0} (f(x_k, q_i) - z_{k,i})^T P_{i0}^{-1} (f(x_k, q_i) - z_{k,i})) \tag{20}$$

$$+\sum_{j=1}^{N_O} \hat{c}_{ij}(f(x_k, o_{k,j}) - z_{k,i})^T P_{ij}^{-1}(f(x_k, o_{k,j}) - z_{k,i})\Bigg]$$

which has no closed form solution due to the nonlinear function $f(\cdot,\cdot)$. Appendix B describes how to compute the solution $x_k^{(t+1)}$ by linearizing $f(\cdot,\cdot)$.

Notice that, since the moving objects are included in the classification (E-Step), they also influence the computation of x_k. Their influence is reflected in the P_{ij} term. When the location of the moving objects is uncertain (due to the prediction of this position, for instance), the value of P_{ij} is high and does not affect the solution.

3.3.4 Updating the map and the moving objects

In this section, we describe how to update the probabilistic grid map and the set of Kalman filters with the last measurements Z_k after convergence of the EM algorithm.

In addition to the maximum likelihood pose of the robot $\hat{x}_k$, the EM also provides an estimate of the values of the correspondence variables c_k. We use this estimate to distinguish between static, dynamic and unknown measurements. Except in those situations where there exist ambiguities, once the robot position is corrected all the weight is assigned to a single source. A simple threshold on the probabilities of the correspondences allows us to classify the measurements in three different sets Z_k^{static}, $Z_k^{dynamic}$ or $Z_k^{unknown}$,

$$z_{k,i} \in \begin{cases} Z_k^{static}, & \text{if } \hat{c}_{i0} > \alpha \\ Z_k^{dynamic}, & \text{if } \sum_{j=1}^{N_O} \hat{c}_{ij} > \alpha \\ Z_k^{unknown}, & \text{otherwise} \end{cases} \tag{21}$$

where the value of the threshold $\alpha > 0.5$ ensures that a measurement only belongs to one of the sets Z^{static}, $Z^{dynamic}$ or $Z^{unknown}$.

The update of the probabilist map is done as in [15], but the process is adapted to deal with the different types of measurements (static,dynamic or unkonwn). On the one hand, all the measurements of a scan contribute to update the free space traversed by their corresponding laser beams (see Figure 4). On the other hand, only those measurements classified as static provide information about the static parts. So as to initialize new static areas, we keep a second grid map with the unknown measurements $Z_k^{unknown}$ in the frontiers of the

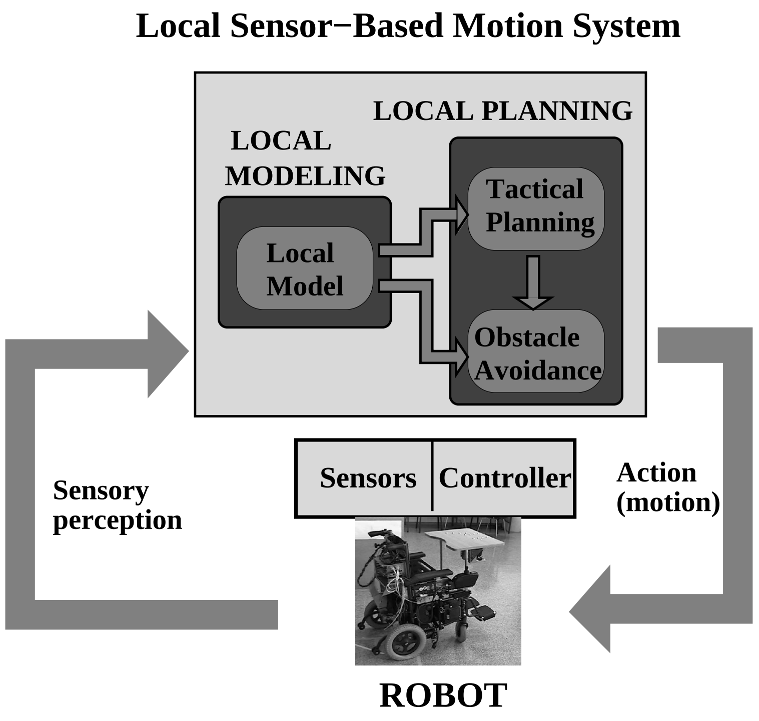


Fig. 5. Overview of the local sensor-based motion system that combines modeling and planning aspects.

explored workspace. If a cell is detected consecutively a given number of time steps, it is included as static in the probabilistic grid map.

In the case of the filters that track the moving objects, we use a segmentation algorithm based on distances to cluster the dynamic measurements $Z_k^{dynamic}$. Then, we use a Joint Probabilistic Data Association [2] scheme to update the set of Kalman filters [3] . So as to deal with new objects, we initialize a filter for those clusters of points that are not assigned to any filter. Furthermore, filters without support are removed after a fixed number of steps.

In summary, we have described in this section an EM algorithm to incrementally compute the maximum likelihood trajectory of the vehicle. Based on this trajectory, the algorithm also computes a map of static obstacles and a map of dynamic obstacles.

4 Integration of the Modeling within the Motion Layer

Local sensor-based motion systems combine modeling and planning aspects. On the one hand, we have proposed in the previous section a technique to model the static and dynamic features of the scenario. On the other hand, the planning aspect in these systems usually combines tactical planning with obstacle avoidance[4]. In this section we outline the tools used in our system [37] and we describe the interactions of the modeling with the rest of the modules in a general framework. We address next the tactical planning and the obstacle avoidance modules.

- **Tactical planning**: computation of the main cruise to drive the vehicle (used to avoid the cyclical motions and trap situations). This module uses the D*Lite planner [24] to compute a path to the goal and to extract the main cruise. The principle of this planner is to locally modify the previous path (available from the previous step) using the changes in the scenario. The module has two different parts: (*i*) the computation of the obstacle changes in configuration space (notice that the grid represents the workspace), (*ii*) the usage of the D*Lite planner over the changes to recompute a path (if necessary). The planner avoids the local minima and is computationally very efficient for real time implementations.
- **Obstacle Avoidance**: computation of the collision-free motion. We chose the Nearness Diagram Navigation [29]. This technique employs a "divide and conquer" strategy based on situations to simplify the difficulty of the navigation. At each time, a situation is selected and the corresponding action computes the motion for the vehicle. This method has been shown to perform well in scenarios that remain troublesome for many existing methods. Furthermore, a technique to take into account the shape, kinematics and dynamic constraints is used to address the local issues of the vehicle [28].

We next describe the interaction between the modules of the system (Figure 5) focusing on the modeling module. Recall that this last module computes both a map of the static structure of the scenario and a map of dynamic obstacles with their locations and velocities.

[3] We could use the final weights provided by the EM algorithm to solve the data association problem between the moving objects and the dynamic measurements. However, this strategy is prone to lose track of the moving objects in the presence of ambiguities [35].

[4] These systems are usually referred as *systems of tactical planning* [42,50,9,30,46,41]

- **Modeling - Tactical planning:** The map of the static structure[5] is the input data to the planner. This is because the role of the tactical planner is to determine at each cycle the main cruise to direct the vehicle. A cruise depends on the permanent structure of the scenario (e.g. walls and doors) and not on the dynamic objects moving around (e.g. people). Notice that using only the static structure overcomes situations that other systems would interpret as trap situations or blocked passages due to the temporal presence of moving objects.
- **Modeling - Obstacle Avoidance:** Both maps are the inputs of the obstacle avoidance technique. While the map of the static structure is directly used as computed by the modeling module, the map of dynamic obstacles is processed to compute an alternative map based on the *predicted collision* point [17]. This new map of obstacles is the other input of the obstacle avoidance. We use a lineal model for the velocities of the robot and the obstacle. For each obstacle i the collision point $p_c^i = (p_{cx}^i, p_{cy}^i)$ is computed by

  $$p_c^i = p_o^i + v_o^i t_c^i \tag{22}$$

  where p_o^i is the obstacle location in the robot reference system[6] and v_o^i is the velocity vector of the obstacle. The collision time t_c^i representes the time when the robot and obstacle i intersect along the motion direction of the robot,

  $$t_c^i = \frac{p_{ox}^i - (R_r + R_o^i)}{v_{r_x} - v_{ox}^i} \tag{23}$$

  where R_r and R_o^i are the radius of the robot and the obstacle respectively.
  Figure 6 illustrates the computation of the predicted collision. Notice that the obstacle avoidance receives a map of predicted collision locations p_c^i. Let us remark that the predicted location of the obstacle depends on the current obstacle location but also on both the vehicle and obstacle relative velocities. Furthermore, if the obstacle moves further away from the robot ($t_c^i < 0$), it is not taken into account. This approach to avoid the moving obstacle implies: (i) if there is a potential collision, the obstacle avoidance method starts the avoidance motion before than if the obstacle was considered static; and (ii) if there is no potential collision, the obstacle is not taken into account.
- **Tactical planning - Obstacle Avoidance:** In the systems of tactical planning, the obstacle avoidance module generates the collision-free motion to align the vehicle toward the cruise computed by the planner. More specifically, the cruise is computed as a subgoal using the direction of the initial

[5] The modeling module also includes those dynamic objects with zero velocity for a predefined period of time in the map of static structure passed to the planner.
[6] The X-axis of the robot reference system is aligned with the instantaneous robot velocity v_r.

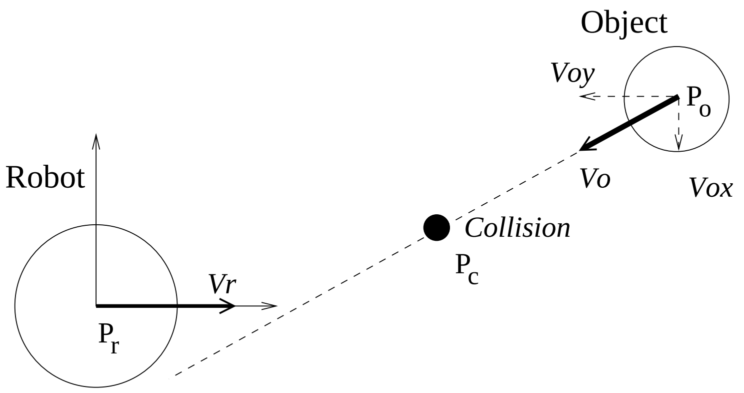


Fig. 6. The object location used by the obstacle avoidance is p_c, which is the predicted collision point according to the current vehicle and object velocities.

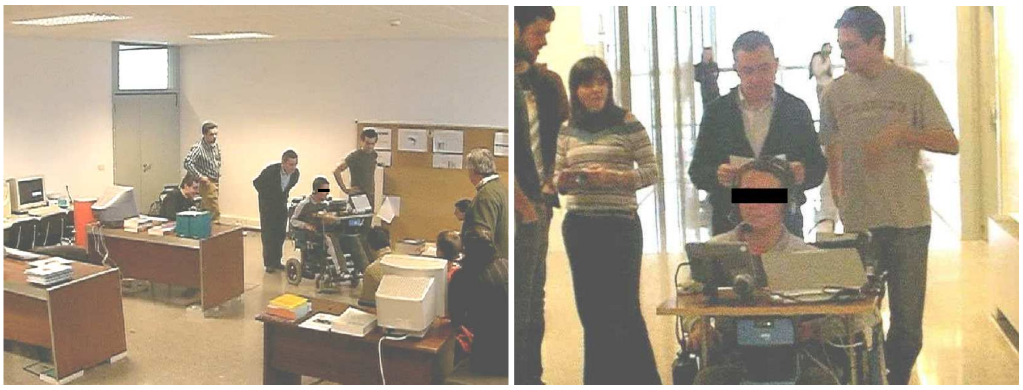
part (predefined distance) of the path.

Globally the system works as follows (Figure 5): given a laser scan and the odometry of the vehicle, the model builder incorporates this information into the existing model. Next, the static and dynamic information of obstacles in the model is selectively used by the planner module to compute the cruise to reach the goal (tactical information). Finally, the obstacle avoidance module uses the planner tactical information together with the information of the obstacles (static and dynamic) to generate the target oriented collision-free motion. The vehicle controller executes the motion and the process restarts with a new sensor measurement. It is important to stress that the three modules work synchronously within the perception - action cycle.

5 Experimental results

This section describes some of the tests that we have carried out to validate the modeling technique (Section 3) and its integration within the motion layer (Section 4). The robot is a commercial wheelchair equipped with two on-board computers and a SICK laser. The vehicle is rectangular $(1.2 \times 0.7 meters)$ with two tractor wheels that work in differential-driven mode. We set the maximum operational velocities to $(v_{max}, w_{max}) = (0.3\frac{m}{sec}, 0.7\frac{rd}{sec})$ due to the application context (human transportation). All the modules work synchronously on the on board $PentiumIII850Mhz$ at the frequency of the laser sensor (5Hz).

We have intensively tested the system with our mobile robot in and out of our laboratory with engineers and with the intended final users of the wheelchair. In this paper, we describe two experiments that we understand will give insight into the performance and benefits of the proposed approach. Firstly, we describe a real run where a child explored our department building. Here, we will focus on the properties of the modeling module. Secondly, we outline a


[redacted] [redacted]

(a) (b)

Fig. 7. Two snapshots of Experiment 1. (a) moving in a office like scenario and (b) traveling along a corridor.

controlled experiment in our laboratory to illustrate the performance of the local sensor motion system.

5.1 Experiment 1: User guided experiment

We describe next a test where a cognitive disabled child drove the vehicle during rush hour at the University of Zaragoza. By using voice commands, he placed goal locations that the motion system autonomously attained. Notice that in this case, the user is responsible for the global aspects of the motion task while the motion system is locally generating the motion.

In the experiment, the child drove the vehicle out of the laboratory (Figure 7a), explored the department (long corridor, Figure 7b) and came back to the initial location without collisions. The time of the experiment was around 20 minutes (including a break of five minutes to calm and relax the children) and the traveled distance was 110 meters. From the motion generation point of view, the performance was very good since the vehicle achieved all the goal locations without collisions. Notice that navigation under these circumstances is not easy since the scenario was unknown and not prepared to move a wheelchair (in many places there was little room to maneuver). In addition, people turned the scenario into a dynamic and unpredictable place and sometimes modified the structure of the environment creating difficult motion situations.

Let us focus on the performance of the modeling module. We implemented the map of static features with a $20m \times 20m$ grid map with a $5cm$ resolution cell centered in the robot location. This spatial domain is large enough to include the goal locations required for the motion. We selected a $5cm$ map resolution since experimentally we observed that it is enough for obstacle avoidance. The size of the map, 400×400, is close to the limit for the planner to comply with

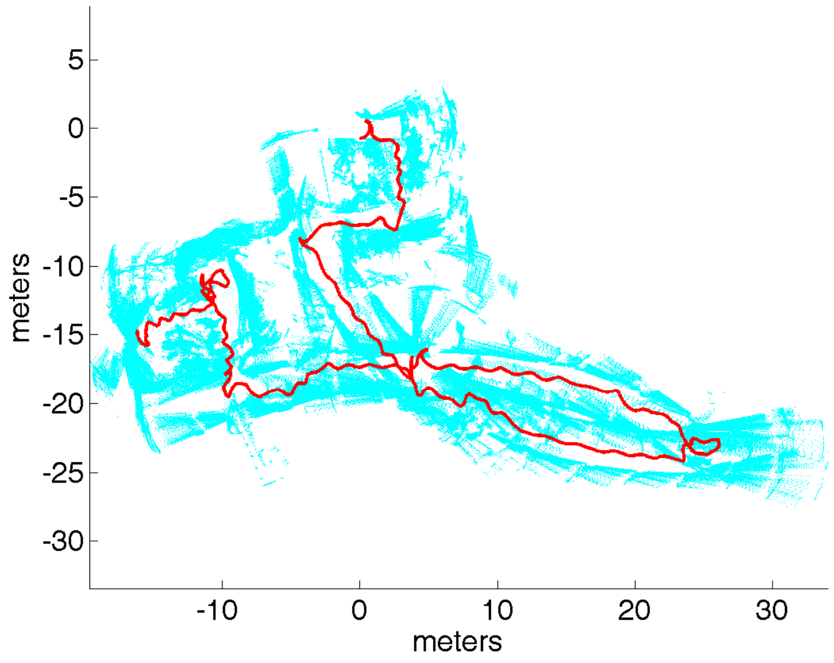


Fig. 8. This Figure shows the raw laser data integrated using the vehicle odometry and the trajectory of the vehicle.

the (worse case) real-time requirements.

Figure 8 shows the raw data of the experiment. From this data, at each point in time, the modeling module computed a map of dynamic objects, including their velocities, and a map of static obstacles. Figure 9 shows the blueprint of the scenario, the trajectory of the vehicle and the models (static and dynamic) computed at two given times in two different places. Notice how despite the odometry errors the local maps of the static structure were correct for the purposes of motion generation. Furthermore, notice how the dynamic objects do not appear in the static map and vice versa. In other words, the static and dynamic structures are separated. There were more than 100 moving objects (people) detected by the system during this experiment. The majority of them corresponded to people moving, however, some of them were false moving objects. These rare situations occurred when the laser beams were almost parallel to the reflected surface and produced specular reflections, or when the beams missed an object because its height was similar to the height at which the laser scan is placed (around $70cm$). In the latter case, due to the motion of the wheelchair, the laser oscillates and sometimes misses the obstacle. As a result, these objects were located in the free space area, classified as dynamic and tracked accordingly.

For real-time operation, it is worth mentioning the low computational requirements of the proposed technique. Figures 10a, b show the number of iterations and the computation time at each step. Despite the clock resolution of the computer ($10msec$), the figures reflect that the cost per iteration is constant.

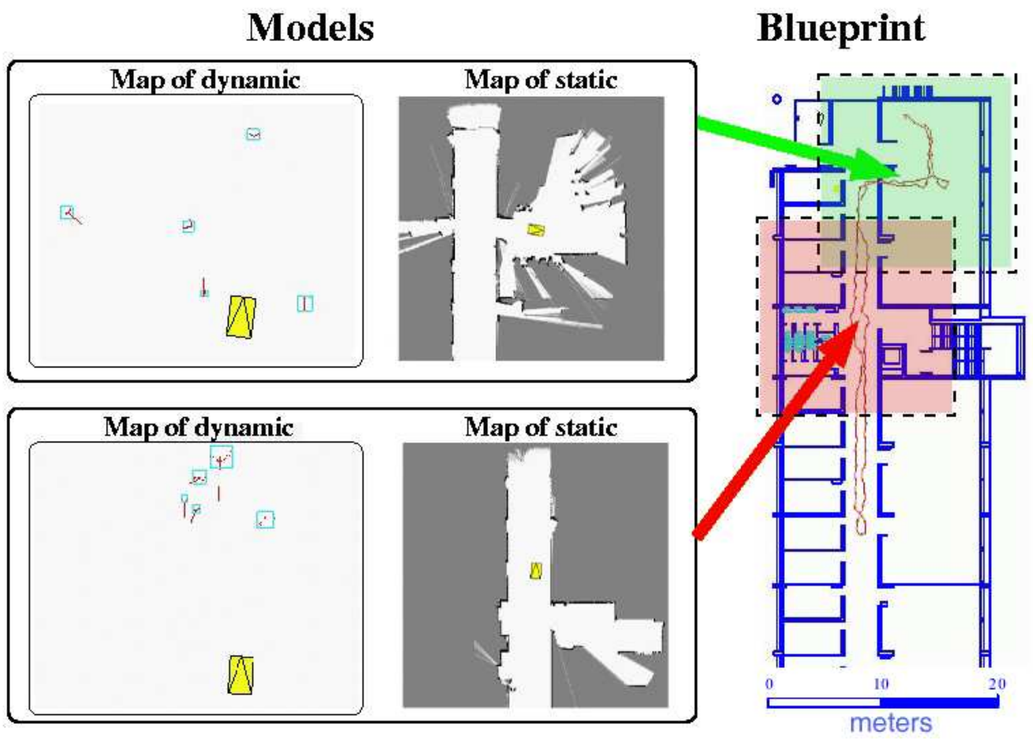


Fig. 9. The right part of the figure shows the trajectory of the experiment within the blueprint of the building. The left part shows the map of moving objects and the map of static obstacles at two different points in time. In the dynamic maps, the rectangles represent the estimated location of the moving objects being tracked containing the observations associated to each of them. The straight lines represent the estimated velocity of the moving object. In the static grid maps, white represents free space, black the obstacles and gray is unknown space.

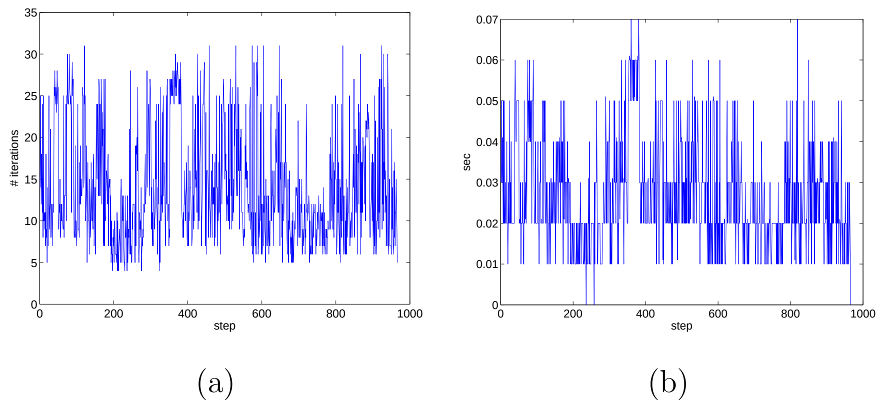
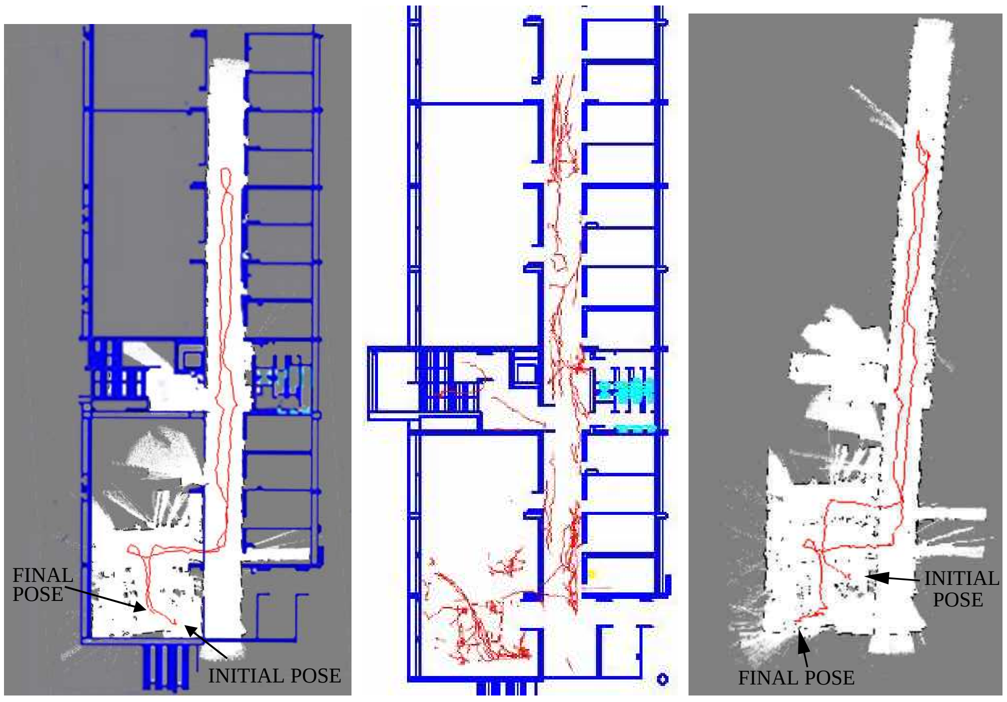


Fig. 10. (a) Number of iterations until convergence for each step of Experiment 2 and (b) the corresponding computation time.

This depends on the number of points of the scan, the grid resolution (number of static points) and the number of moving objects. The mean values for the whole process are 14.1 iterations and $21msec$. The time spent in the update of the map and in the prediction and update of the filters is negligible and is



(a) (b) (c)

Fig. 11. (a) Map of the static structure, the robot trajectory and the blueprint of the building (notice that the architect original plan was modified during construction). (b) Trajectories of the dynamic obstacles and the blue print. (c) Map obtained without taking into account the moving objects. Note how the dynamic objects affect the displacement estimation along the corridor. As a result, the estimated final vehicle location is far from the original one and the map is corrupted.

below the clock resolution. The proposed method provides a fast solution to the local modeling problem, which is important to integrate it with the other modules of the architecture for real-time operation.

Although it is beyond the application and context of the paper, we understand that it is interesting to discuss the performance of the technique facing larger scenarios. To test this situation, we processed offline the previous dataset (Figure 8) but using a grid map of $80m \times 80m$ to represent the whole area covered by the experiment. In this mapping context, this experiment is not easy since the laboratory (first room) is very unstructured due to the presence of chairs, tables and other people; and the corridor is very large and does not contain much information in the direction of the corridor (which makes it difficult to correct the robot displacement in the presence of moving people).

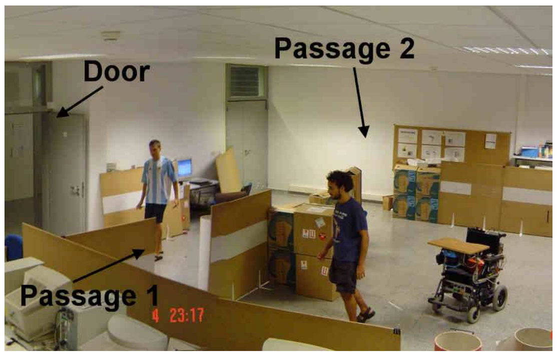
The performance of the mapping technique was very good since it was still able to separate the 100 dynamic obstacles from the static structure (map of static parts) and thus they did not affect the estimated vehicle pose. Figures 11a and b show the final map of the static structure and the trajectories of the dynamic objects tracked with the reference of the blueprint. Notice that the map of static objects fits perfectly with the blueprint, which is an indicator of the quality of the map. In the bottom part of the blueprint, the trajectories go through a wall. This is because the final building was modified after the architect did the plan. Furthermore, the map of the dynamic structure shows 100 trajectories that correspond to the people that moved around. Notice how all the trajectories are in free space which is also a good indicator for the dynamic map quality.

To check the influence of the dynamic objects, we used our method without considering the moving objects (i.e. considering all the measurements as static). The results strongly affected the resulting map not only in the motion along the corridor, but also in the other directions. For instance, the corridor of Figure 11c is slightly curved due to orientation errors accumulated when exploring it and the final location error is big. This result is consistent with the difficulty that many researchers have reported related with map building in the presence of dynamic obstacles [21] and stresses the advantage of the proposed technique.

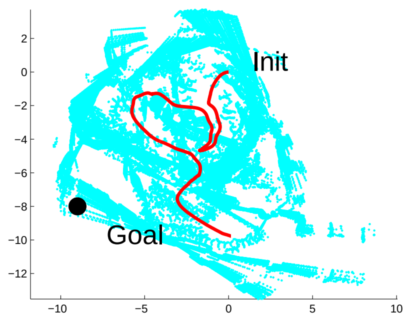
5.2 Experiment 2: Fully autonomous experiment

We next describe a more academic experiment to derive conclusions for the local sensor-based motion system but focusing on navigation performance. The objective of the experiment was to get the wheelchair out of the laboratory (Figure 12a). All the scenario was initially unknown and only the target location was given in advance to the system. Initially, the vehicle proceeded toward the *Passage* 1 avoiding collisions with the people that move around. Then, we blocked the passage creating a global trap situation that was detected by the system. The vehicle moved backwards through *Passage* 2 and then traversed the *Door* exiting the room and reaching the goal location without collisions. The time of the experiment was $200sec$ and the distance traveled around $18m$.

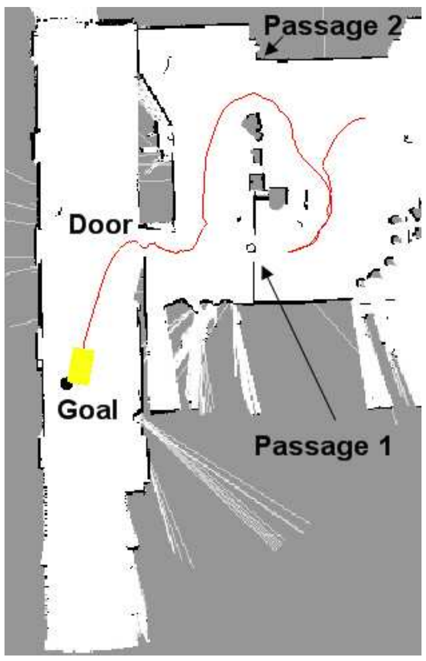
As in the previous example, the performance of the modeling module was good enough for the other modules of the architecture. Figure 12b shows the raw laser data using the odometry readings and Figure 12c shows the final map produced by the modeling module when the vehicle reached the goal location. The trajectories of the moving objects tracked during the experiment are shown in Figure 12d. Most of them correspond to people walking in the free space of the office. There were also some false positives due to misclassification that occurred mainly in the same situations as in Experiment 1. Regarding navigation, the motion system reliably drove the vehicle to the final location without collisions. Recall that the maps generated by the modeling module are the basis for planning and obstacle avoidance. In general, a rough model



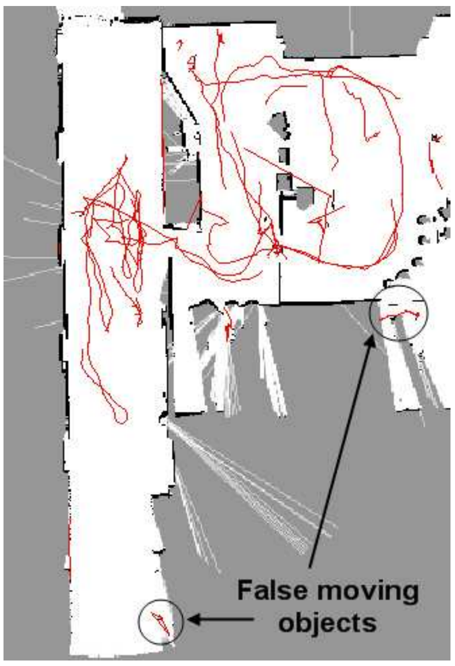
(a)



(b)



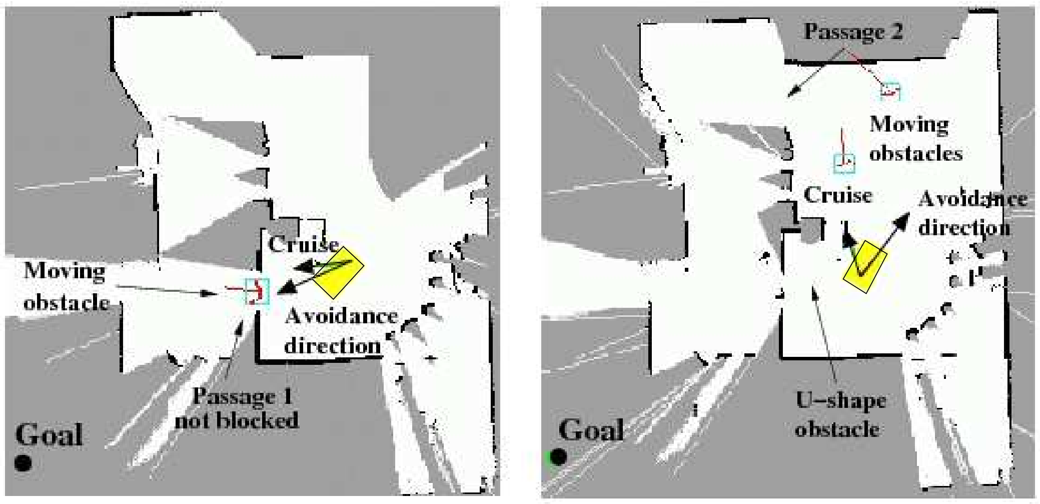
(c)



(d)

Fig. 12. (a) Snapshot of Experiment 2. The objective was to drive the vehicle out of the office through the *Door*. (b) Real laser data and trajectory of Experiment 2 using the raw odometry readings. (c) The map built during the experiment and the vehicle trajectory. The map shows the occupancy probability of each cell. White corresponds to a probability of zero (free cell) and black to a probability of one (occupied cell). (d) The trajectories of the detected moving objects.

or only odometry readings are not enough and likely would lead to navigation mission failures. The quality of the model and the localization is specially relevant to avoid obstacles no longer perceived with the sensor due to visibility constraints; to deal with narrow passages where accumulated errors can block the passage even if there is enough space to maneuver; or to approach the vehicle to the desired final position with enough precision. All these situations were correctly managed due to the quality of the models generated with the proposed modeling technique.



(a) (b)

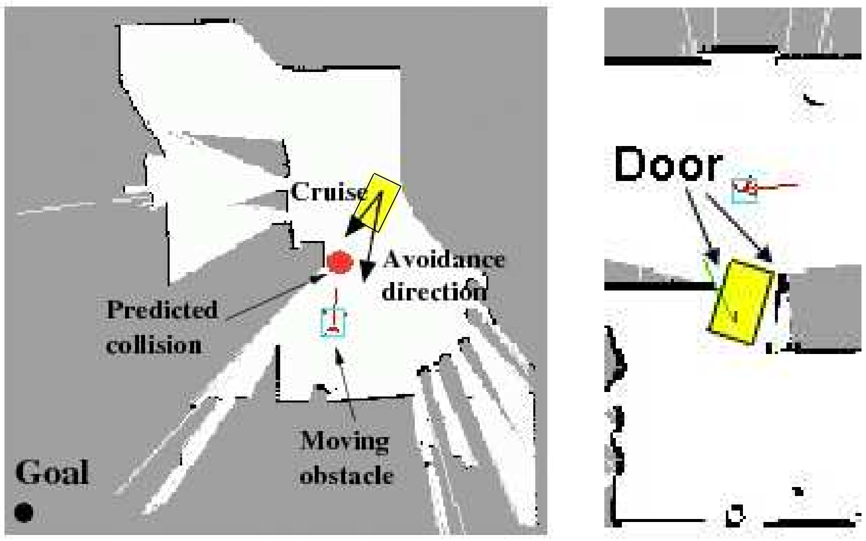
Fig. 13. (a) A moving obstacle placed in the area of passage and (b) robot avoiding a trap situation. The figures show the tracked moving objects (rectangles), the dynamic observations associated to them and the estimated velocities. The two arrows on the vehicle show the cruise computed by the planner module and the direction of motion computed by the obstacle avoidance one.

We next describe several situations where the selective use of the static and dynamic information improved the motion generation.

The planner computed at every point in time the tactical information needed to guide the vehicle out of the trap situations (the cruise) using only the static information. The most representative situations happened in the *Passage* 1. While the vehicle was heading along this passage, people were crossing it. However, since the humans were tracked and labeled dynamic they were not used by the planner and thus the cruise pointed toward this passage (Figure 13a) and the vehicle aligned with this direction. Notice that systems that do not model dynamic obstacles would consider the human static and the vehicle trapped within a U-shape obstacle[7]. Next, a human placed an obstacle in the passage when the vehicle was about to reach it. The vehicle was trapped in a large U-shape obstacle. After a given period, the modeling module included this obstacle in the static map passed to the planner. Immediately, the planner computed a cruise that pointed toward the *Passage* 2. The vehicle was driven toward this passage avoiding the trap situation (Figure 13b).

The obstacle avoidance module computed the motion taking into account the geometric, kinematic and dynamic constraints of the vehicle [28,31]. The method used the static information included in the map and also the predicted collision locations of the objects computed using the obstacle velocities. Figure

[7] This situation is similar to the situation depicted in Figure 1b



(a) (b)

Fig. 14. (a) Moving obstacle going toward the robot. The figure shows the tracked moving objects (rectangles), the dynamic observations associated to them and the estimated velocities. The two arrows on the vehicle show the cruise computed by the planner module and the direction of motion computed by the obstacle avoidance one. (b) Detail of the robot maneuver to cross the *Door*.

14a depicts an object moving toward the robot, and how the predicted collision creates an avoidance maneuver. Note that, although the Nearness Diagram does not consider dynamic objects, the predicted collision location allows it to anticipate the maneuver. Furthermore, obstacles that move further away from the robot are not considered. In Figure 13b the two dynamic obstacles were not included in the avoidance step (whereas systems that do not model the dynamic objects would consider them).

The performance of obstacle avoidance module was determinant in some circumstances, especially when the vehicle was driven among very narrow zones. For example, when it crossed the door (Figure 14b), there were less than $0.1m$ at both sides of the robot. The movement computed by the obstacle avoidance module was free of oscillations and, sometimes, was directed toward zones with great density of obstacles or far away from the final position. All the robot constraints were considered by the obstacle avoidance method generating feasible motions in the different situations. That is, the method achieved robust navigation in difficult and realistic scenarios.

In summary, the modeling module was able to model the static and dynamic parts of the environment. The selective use of this information allows the planning and obstacle avoidance modules to avoid the undesirable situations that arise from false trap situations and improve the obstacle avoidance task.

Furthermore, the integration within the architecture allows to fully exploit the advantages of hybrid sensor-based navigation systems that perform in difficult scenarios avoiding typical problems such as trap situations.

6 Discussion and Conclusions

In this paper we have addressed two issues of great relevance in local sensor-based motion: how to model the static and dynamic parts of the scenario and how to integrate this information with a local sensor-based motion planning system.

Regarding the modeling aspect, most of previous works [45,51,52,21] assume a known classification within the optimization process. This means that the classification is done prior to the estimation of the vehicle location. They focus on the reliable tracking of the moving objects or on the construction of accurate maps. The algorithm proposed in [22] iteratively improves the classification via an EM algorithm. This is a batch technique that focuses on the detection of spurious measurements to improve the quality of the map. The approach presented in [33,34] applies learning techniques but does not improve the vehicle location and does not use probabilistic techniques to track the moving objects. Our contribution in the modeling aspect is to incorporate the information about the moving objects within a maximum likelihood formulation of the scan matching process. In this way, the nature of the observation is included in the estimation process. The result is an improved classification of the observations that increases the robustness of the algorithm and improves the robot pose estimation, the map and the moving objects location.

However, the drawback of this type of techniques is that they do not consider the uncertainties and the corresponding correlations of the robot poses, the map and the moving objects. Moreover, the set of poses is fixed and cannot be modified in subsequent steps. This represents a problem when closing loops if the accumulated error is big. In the case of sensor based navigation, the spatial domain of the problem is small and, thus, allows us to obtain enough accurate models for real-time operation.

In any case, all the mapping methods assume a hard classification between static and dynamic objects. This is clearly a simplification of the real world since there are objects that can act as static or dynamic; for instance, doors, chairs, tables, cars, etc. Although there exist some preliminary work on the estimation of the state of some of these objects [47,6], we believe a prior on the behavior of the objects will greatly simplify the problem. This can be done using another type of sensors as cameras or 3D range sensors.

The second issue is the integration of the modeling module in a local sensor-based motion system taking advantage of the dynamic and static information. The system selectively uses this information in the planning and obstacle avoidance modules. As a result, many problems of existing techniques (that only address static information) are avoided without sacrificing the advantages of the full hybrid sensor-based motion schemes. Notice that the planning - obstacle avoidance strategy relies on the information provided by the modeling module. Since our model assumes a constant lineal velocity model, the predicted collision could not be correct when this assumption does not hold. However, this effect is mitigated since the system works at a high rate rapidly reacting to the moving obstacle velocity changes.

One thing to remark is that the local planning strategy is an approximation of the full motion-planning problem with dynamic obstacles (recall that this problem is NP-hard in nature). In this paper we have addressed it with a hybrid system made up of a tactical planning module and an obstacle avoidance technique (a simplification). Although there exist reactive techniques that are designed to explicitly deal with dynamic obstacles [16,18,40], we have selected one that does not account for this information (this is the reason why we use the collision prediction concept). The selection of the reactive techniques is a trade off between performance facing very dynamic scenarios or places where it is very difficult to maneuver. This is because it is well known that the techniques that address the motion planning under dynamic obstacles are conservative in the motion search space (losing maneuvrability in constrained spaces). In our case, due to the wheelchair application, we used a method designed to maneuver in environments with little room to maneuver (such as doors or narrow corridors) and we improved the behavior in dynamic situations with the collision prediction concept. However, for other applications, nothing prohibits the use of other method in the proposed framework.

The experimental results confirm that the modeling method is able to deal with dynamic environments and provide enough accurate models for sensor-based navigation. The integration with a sensor-based planner system allows to drive the vehicle in unknown, dynamic scenarios with little space to maneuver. The system avoids the typical trap situations found under realistic operation and, in particular, those created by moving obstacles.

References

[1] R.C. Arkin. *Behavior-Based Robotics.* The MIT Press, 1999.

[2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Mathematics in Science and Engineering. Academic Press., 1988.

[3] Y. Bar-Shalom, XR Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation.* J. Wiley and Sons, 2001.

[4] O. Bengtsson and A-J. Baerveldt. Localization in changing environments by matching laser range scans. In *EURobot*, pages 169–176, 1999.

[5] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[6] Peter Biber and Tom Duckett. Dynamic maps for long-term operation of mobile service robots. In *Robotics: Science and Systems*, June 8-10 2005.

[7] Peter Biber and Wolfgang Strafler. The normal distributions transform: A new approach to laser scan matching. In *IEEE Int. Conf. on Intelligent Robots and Systems*, Las Vegas, USA, 2003.

[8] S. Blackman and R. Popoli. *Design and analysis of modern tracking systems.* Artech House, Norwood, MA, 1999.

[9] O. Brock and O. Khatib. High-Speed Navigation Using the Global Dynamic Window Approach. In *IEEE Int. Conf. on Robotics and Automation*, pages 341–346, Detroit, MI, 1999.

[10] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *In Proc. of the 27th Annual IEEE Symposium on the Foundations of Computer Science*, pages 49–60, 1987.

[11] J. A. Castellanos and J. D. Tardós. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach.* Kluwer Academic Publishers, Boston, 1999.

[12] P. Cheeseman and P.Smith. On the representation and estimation of spatial uncertainty. *International Journal of Robotics*, 5:56–68, 1986.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. 34:1–38, 1977.

[14] A. Doucet, S.J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[15] A. Elfes. Occupancy grids: A probabilistic framework for robot perception. *PhD thesis*, 1989.

[16] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. Journal of Robotic Research*, 17(7):760–772, 1998.

[17] A. Foisy, V. Hayward, and S. Aubry. The use of awareness in collision prediction. In *IEEE Int. Conf. on Robotics and Automation*, 1990.

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

[19] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Conference on Intelligent Robots and Applications (CIRA)*, Monterey, CA, 1999.

[20] D. Hähnel. Mapping with mobile robots. *PhD thesis, University of Freiburg*, 2004.

[21] D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[22] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[23] B. Jensen and R. Siegwart. Scan alignment with probabilistic distance metric. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[24] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *International Conference on Robotics and Automation*, Washington, USA, 2002.

[25] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[26] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Intelligent and Robotic Systems*, 18:249–275, 1997.

[27] G. McLachlan and T. Krishnan. *The EM algorithm and extensions.* Wiley series in probability and statistics. J. Wiley and Sons, 1997.

[28] J. Minguez and L. Montano. Robot Navigation in Very Complex Dense and Cluttered Indoor/Outdoor Environments. In *15th IFAC World Congress*, Barcelona, Spain, 2002.

[29] J. Minguez and L. Montano. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.

[30] J. Minguez and L. Montano. Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems*, 52(4):290–311, 2005.

[31] J. Minguez, L. Montano, and J. Santos-Victor. Abstracting the Vehicle Shape and Kinematic Constraints from the Obstacle Avoidance Methods. *Autonomous Robots*, 20(43-59), 2006.

[32] J. Minguez, L. Montesano, and F. Lamiraux. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Transactions on Robotics*, 22(5):1047–1054, 2006.

[33] J. Modayil and B. Kuipers. Bootstrap learning for object discovery. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-04)*, Sendai, Japan, 2004.

[34] J. Modayil and B. Kuipers. Towards bootstrap learning for object discovery. In *AAAI-2004 Workshop on Anchoring Symbols to Sensor Data*, 2004.

[35] L. Montesano. Detection and tracking of moving objects from a mobile platform. application to navigation and multi-robot localization. *PhD thesis, Universidad de Zaragoza*, 2006.

[36] L. Montesano, J. Minguez, and L. Montano. Probabilistic scan matching for motion estimation in unstructured environments. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[37] L. Montesano, J. Minguez, and L. Montano. Lessons learned in integration for sensor-based robot navigation systems. *International Journal of Advanced Robotic Systems*, 3(1):85–91, 2006.

[38] H. P. Moravec and A. Elfes. High Resolution Maps from Wide Angle Sonar. In *IEEE Int. Conf. on Robotics and Automation*, pages 116–121, March 1985.

[39] K. Murphy. Dynamic bayesian networks: Representation, inference and learning. *PhD thesis, UC Berkeley, Computer Science Division*, 2002.

[40] E. Owen and L. Montano. Motion planning in dynamic environments using the velocity space. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB (CA), August 2005.

[41] R. Philipsen and R. Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, 2003.

[42] S Ratering and M. Gini. Robot navigation in a known environment with unknown moving obstacles. In *International Conference on Robotics and Automation*, pages 25–30, Atlanta, USA, 1993.

[43] C. Schlegel. *Navigation and Execution for Mobile Robots in Dynamic Environments - An Integrated Approach.* PhD thesis, University of Ulm, 2004.

[44] D. Schulz, W. Burgard, D. Fox, and A. Cremers. Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association. In *IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001.

[45] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 22(2):99–116, 2003.

[46] C. Stachniss and W. Burgard. An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 508–513, Switzerland, 2002.

[47] C. Stachniss and W. Burgard. Mobile robot mapping and localization in non-static environments. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, Pittsburgh, PA, USA, 2005.

[48] S. Thrun, W. Burgard, and D. Fox. A real time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *IEEE Int. Conf. on Robotics and Automation*, San Francisco, USA, 2000.

[49] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT press, 2005.

[50] I. Ulrich and J. Borenstein. VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In *IEEE Int. Conf. on Robotics and Automation*, pages 2505–2511, San Francisco, USA, 2000.

[51] C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *IEEE Int. Conf. on Robotics and Automation*, Washington, USA, 2002.

[52] C.-C. Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

A Likelihood models

This appendix describes the computation of the mean and covariance of the Gaussian likelihood models $p_s(.)$ and $p_d(.)$ of Section 3.3.1. So as to obtain an analytical expression, we assume Gaussian uncertainties in the robot pose and in the location of the static and dynamic objects and Gaussian noise in the measurement process,

$$x \sim N(x^{true}, P) \tag{A.1}$$
$$q \sim N(q^{true}, Q) \tag{A.2}$$
$$z \sim N(z^{true}, R) \tag{A.3}$$

Note that here we use q as a generic correspondence point for the measurement z. Although the computations are the same for static and dynamic correspondences, the covariance matrix for each type of association is different. We use a fixed covariance matrix for grid cells and the uncertainty of the Kalman filter prediction for moving objects.

The function $f(x, q)$ is the transformation of the point $q = (q_x\ q_y)^T$ through the relative location $x = (t_x, t_y, \theta)^T$,

$$f(x, q) = \begin{pmatrix} \cos\theta q_x - \sin\theta q_y + t_x \\ \sin\theta q_x + \cos\theta q_y + t_y \end{pmatrix} \tag{A.4}$$

Linearizing the function $f(x,q)$ using a first order Taylor series approximation, we define the likelihood term as

$$p(z \mid x,q) = \int\int \underbrace{p(z \mid x,q)}_{N(f(x,q),R)} \underbrace{p(x)}_{N(x,P)} \underbrace{p(q)}_{N(q,Q)} \, dxdq = N(z; f(x,q), C) \tag{A.5}$$

where the covariance matrix C is

$$C = R + J_x P J_x^T + J_q Q J_q^T \tag{A.6}$$

The matrices J_x and J_q are the Jacobians of $f(x,q)$ with respect to x and q evaluated at the current estimates,

$$J_x \equiv \left.\frac{\partial f(x,q)}{\partial x}\right|_{x,q} = \begin{pmatrix} 1 & 0 & -q_x \sin\theta - q_y \cos\theta \\ 0 & 1 & q_x \cos\theta - q_y \sin\theta \end{pmatrix}$$

$$J_q \equiv \left.\frac{\partial f(x,q)}{\partial q}\right|_{x,q} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

In addition to this, using the function $f(\cdot,\cdot)$ we can define the Mahalanobis distance between z and q to select the appropriate static obstacle from the grid map

$$D_M^2(z,q) = [f(x,q) - z]^T C^{-1} [f(x,q) - z] \tag{A.7}$$

B M-Step minimization

This appendix addresses the minimization of the function

$$Q(x_k, x_k^{(t)}) = \sum_{i=1}^{N_z} \Bigg[\hat{c}_{i0} (f(x_k, q_i) - z_{k,i})^T P_{i0}^{-1} (f(x_k, q_i) - z_{k,i})) + \sum_{j=1}^{N_O} \hat{c}_{ij} (f(x_k, o_{k,j}) - z_{k,i})^T P_{ij}^{-1} (f(x_k, o_{k,j}) - z_{k,i}) \Bigg] \tag{B.1}$$

Due to the nonlinear function $f(\cdot,\cdot)$, one should use an iterative method to minimize $Q(x_k, x_k^{(t)})$. However, since the correspondences change at each iteration of the EM algorithm, we rather use a single iteration to improve the current estimate. This is known as generalized EM [27] and the algorithm still converges to the local minimum.

Based on the linearization of $f(\cdot,\cdot)$ presented in appendix A, the estimate of the parameter vector x^{LS} is,

$$x^{LS} = [H^T C^{-1} H]^{-1} H^T C^{-1} E \tag{B.2}$$

where the matrices E and H are formed by the contributions of each measurement $z_{k,i}$ to the function $Q(x_k, x_k^{(t)})$

$$H = \begin{bmatrix} H_1 \\ \vdots \\ H_{N_z} \end{bmatrix} \qquad E = \begin{bmatrix} E_1 \\ \vdots \\ E_{N_z} \end{bmatrix} \qquad C = \begin{bmatrix} C_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_{N_z} \end{bmatrix}$$

with

$$H_i = \begin{bmatrix} J_x(x^{(t)}, q_i) \\ J_x(x^{(t)}, o_{k,1}) \\ \vdots \\ J_x(x^{(t)}, o_{k,N_O}) \end{bmatrix} ; E = \begin{bmatrix} -(f(x_k^{(t)}, q_i) - z_{k,i}) + J_x(x^{(t)}, q_i)x^{(t)} \\ -(f(x_k^{(t)}, o_{k,1}) - z_{k,i}) + J_x(x^{(t)}, o_{k,1})x^{(t)} \\ \vdots \\ -(f(x_k^{(t)}, o_{k,N_O}) - z_{k,i}) + J_x(x^{(t)}, o_{k,N_O})x^{(t)} \end{bmatrix}$$

$$C_i = \begin{bmatrix} \frac{1}{\hat{c}_{i0}} P_{i0} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\hat{c}_{iN_O}} P_{iN_O} \end{bmatrix} \tag{B.3}$$

with N_z is the number of measurements and N_O is the number of moving objects.