Modeling the Static and the Dynamic Parts of the Environment to Improve Sensor-based Navigation

Luis Montesano Javier Minguez Luis Montano
Instituto de Investigación en Ingeniería de Aragón
Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
{montesano,minguez,montano}@unizar.es

***Abstract*— This paper addresses the modeling of the static and dynamic parts of the scenario and how to use this information within a real sensor-based navigation system. The contribution in the modeling aspect is a formulation of the *Detection and Tracking of Mobile Objects* and the *Simultaneous Localization and Map Building* in such a way that the nature (static/dynamic) of the observations is included in the estimation process. This is achieved by a set of filters tracking the moving objects and a map of the static structure constructed on line. In addition, this paper discusses how this modeling module is integrated in a real sensor-based navigation 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.**

I. Introduction

Currently, the vehicle motion in unknown and dynamic environments is computed by hybrid architectures that combine aspects of modeling, tactical planning and obstacle avoidance. The skill to model the environment distinguishing the dynamic and static parts opens a new dimension in these systems, since it allows a selective treatment of this information to improve the performance of all the modules of the architecture. This greatly ameliorates the overall behavior of the sensor-based navigation system. In this work we present a modeling module that includes the detection and tracking of moving objects, and its integration within the navigation architecture that currently works on our wheelchair vehicle.

The first key aspect of our system is to model the environment. Modeling the static parts of the environment from a mobile robot is known as the Simultaneous Localization and Map Building (SLAM) and it has been an active research area during the last years [4], [17]. On the other hand the Detection And Tracking of Moving Objects (DATMO) is also a well-studied problem [3], [16]. However, a robust solution requires to perform both tasks at the same time.

Most of the DATMO-SLAM systems use the sensor information to model the static parts of the environment with a map and apply some filtering techniques to track the moving objects. For example [7] use a feature based approach to detect the moving objects in the range information. Next, they use Joint Probabilistic Data Association particle filters to track the moving objects and a probabilistic SLAM technique to build the map. In [18] a rigorous formulation of the DATMO-SLAM problem is provided assuming a classification of the observations into static and dynamic. Their detection algorithm is based on the violation of the free space and the tracking on extended Kalman filters (EKF). Another technique detects and filters dynamic measurements while solving the SLAM problem using the Expectation Maximization algorithm [8]. It classifies the observations during the expectation step and correct the robot pose in the maximization step. However they do not take into account the uncertainty of the robot motion in the classification step. Thus difficulties arise in the presence of large odometry errors if many observations are labeled dynamic.

The second key aspect is how the modeling module is integrated within a motion architecture. Most of the hybrid motion systems that work in unknown and dynamic scenarios assume that all the sensory information available is instantaneously static (there is no dynamic obstacle modeling) [1], [2], [14]. These systems perform very well in most surroundings due to a combination of environment modeling (short term memory), tactical planning (that allows to avoid the potential minima and the trap situations) and the obstacle avoidance (that computes the motion using the information of the previous modules). Nevertheless, there exist situations in which the behavior of these systems is degraded due to the necessity to model the dynamic part of the environment. Significant examples are an obstacle moving in a direction where it will collide with the robot, or people that temporarily obstruct zones of passage like doors or corridors.

On the other hand there are systems that model both, the static and dynamic parts of the environment. This information is used next to improve the vehicle location and the avoidance of obstacles [9], [15]. Nevertheless, the limitation of these systems lies in the lack of an integration architecture that solves problems associated with navigation. For example they fall in trap situations or cyclic behaviours, or they exhibit difficulties to compute safe motion in troublesome scenarios (very dense, cluttered and complex).

A reliable solution must address both: a module able to model the static and dynamic parts of the scenario, and the integration within an architecture able to deal with the typical navigation issues. In fact these are the two contributions of this work. The first is a modeling module that carries out DATMO and SLAM at the same time. Our formulation extends the work of [18] to jointly classify the nature of the observations and solve the SLAM problem. The second contribution is the integration of this module in the architecture. The usage of the static and dynamic information selectively by the planning and obstacle avoidance modules allows to avoid the undesirable situations outlined previously, while fully exploiting the advantages of an hybrid sensor-based navigation system. We present experimental results with a wheelchair vehicle working in a realistic scenario.

The rest of the paper is organized as follows. Next section briefly introduces our architecture. Section III and IV describe our method to model dynamic environments and section V shows how this information is used in our system. Experimental results are presented in section VI.

II. Navigation System

We give in this Section a global vision of the navigation system architecture [14]. It integrates three modules with the following functionalities: model construction, tactical motion planning and obstacle avoidance,

- **Model Builder Module**: construction of a model of the static and dynamic parts of the environment. We describe the static parts with a probabilistic occupancy grid map (with limited size and that travels centered with the robot), and track the moving objects with a set of filters. This model increases the spatial domain of the planning and is used as local memory for the obstacle avoidance. Section III provides a detailed description of this module.
- **Planner Module**: extraction of the connectivity of the free space (used to avoid the cyclical motions and trap situations). We developed a planner that computes the existence of a path that joins the robot and goal locations [14]. The planner constructs iteratively a graph whose nodes are locations in the space and the arcs are tunnels of free space that joins them. When the goal is reached, the current tunnel contains a path to the goal. This planner avoids the local minima and is very efficient so that it can be executed in real time.
- **Obstacle Avoidance Module**: computation of the collision-free motion. We chose the Nearness Diagram Navigation (ND method in short) [13], which is based on selecting at each time a navigational situation and to apply a motion law adapted to each one. In other words it employs a "divide and conquer" strategy based on situations to simplify the difficulty of the navigation. This method has demonstrated to perform in scenarios that remain troublesome for many existing methods.

Globally the system works as follows (Figure 1): 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 course to follow 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 motion is executed by the vehicle controller and the process restarts with a new sensorial measurement. It is important to stress that the three modules work synchronously within the perception - action cycle. Next, we address the two main issues of this paper: the model builder module and how its information is employed by the other modules.

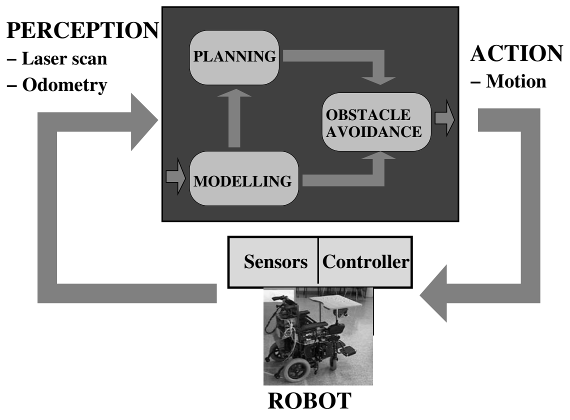


Fig. 1. Overview of the sensor-based navigation system

III. Modeling dynamic environments

Let $z_k = \{z(1)_k, ..., z(m)_k\}$ be the m observations obtained by the robot at time k and u_k the motion command executed at time k. The sets $Z_k = \{z_1, ..., z_k\}$, $U_k = \{u_0, ..., u_k\}$ represent the observations and motion commands up to time k. We denote the robot location at time k by x_k and the map of the static environment by M. Let $O_k = \{o_1, ...o_n\}$ be the location of the moving objects at time k. The objective is to estimate the poterior distribution $p(O_k, x_k, M \mid Z_k, U_{k-1})$ at each point in time.

Previous work on DATMO-SLAM [18] factorizes the distribution assuming that the observations are classified into static and dynamic, $z_k = z_k^s + z_k^d$. As a result classical techniques are used to solve the SLAM problem and the tracking of moving objects independently. In real applications this classification is usually not available. One can classify the points at each time step before estimating $p(O_k, x_k, M \mid Z_k, U_{k-1})$ using the information of the previous step or some feature based approach. However, this represents a hard decision and, consequently, a misclassification at this step will not be corrected.

In this paper we explicitly include the classification process within our formulation. We define a binary variable $c_k = \{c_k(1), ...c_k(m)\}$ to represent the nature of the observation, $c_k(i) = 1$ if the observation has been generated by a static object and $c_k(i) = 0$ if not. We need to estimate c_k together with the pose x_k, the map M and the location of the objects O_k,

$$p(O_k, x_k, M, c_k \mid Z_k, U_{k-1}) \propto p(z_k, c_k \mid O_k, x_k, M) \quad (1)$$
$$p(O_k \mid x_k, M, Z_{k-1}, U_{k-1}) p(x_k, M \mid Z_{k-1}, U_{k-1})$$

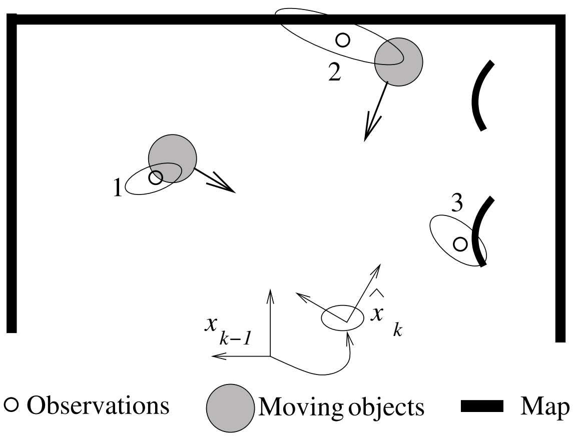


Fig. 2. Uncertain location of observations with respect to the model of the environment. The ellipses associated to each observation represent their uncertainty due to the current uncertain location of x_k with respect to x_{k-1}. This motion uncertainty affects differently to each point. The classification step integrates over all the possible locations of the observations.

The derivation of the previous equation uses the Bayes theorem and conditional probabilities together with the Markov assumption. Due to the numerous variables involved the full derivation is not included to keep the paper readable. According to Eq. 1 the process requires to evaluate the likelihood term $p(z_k, c_k \mid O_k, x_k, M)$ which also includes the classification variables. The terms $p(O_k \mid x_k, M, Z_{k-1}, U_{k-1})$ and $p(x_k, M \mid Z_{k-1}, U_{k-1})$ represent the evolution of the robot location x_k, the map M and the moving objects O_k.

IV. Model builder implementation

In this section we describe the implementation of the model builder module according to the formulation of the previous section. Our particular implementation is designed to integrate it within the navigation architecture of Section II which is running on a wheelchair equipped with a 2D laser range finder. The map is implemented as a probabilistic grid map. It travels centered with the robot and has a limited size. The information contained in the grid map is enough to perform sensor-based navigation and implicitly forgets those areas far away from the current position of the robot. On the other hand, we use a set of independent extended Kalman filters to track the detected moving objects.

A. Joint classification and pose estimation

We use a maximum likelihood incremental approach ([18], [7]) to estimate the relative pose $\hat{x}_k$, the map $\hat{M}_k$ and the moving objects $\hat{O}_k$. Maximizing $p(z_k, c_k \mid M, x_k, O_k)$ is a hard task which involves continuous and discrete variables. We present here an extension to the IDC algorithm [11] proposed by Lu and Milios that jointly estimates the position of the robot and classify the observations into static or dynamic. The classical IDC algorithm computes the relative motion of the robot between two scans. The algorithm iterates over two steps until it converges. First it computes a set of correspondences between the points of the reference scan and the new scan. Then it improves the robot pose estimation using a least squares minimization of the errors of these correspondences.

We want to find the $\hat{x}_k$ and $\hat{c}_k$ that maximizes the likelihood term. To cope with the classification we add a new step to the IDC algorithm. At each iteration before computing the correspondences we classify the points using the current estimated robot pose, the estimated map and the estimated locations of the moving objects. We then estimate the correspondences for those points classified as static and compute $\hat{x}_k$. As in the IDC the algorithm iterates until it converges.

It remains to describe the classification procedure. Our main purpose is to provide a robust classification that considers all the information available at the previous step, i.e. the static map and the tracked moving objects, together with the uncertainty of the robot position due to its last displacement.

For each observation $z_k(i)$ we compute the probability of the classification variable $c_k(i) = 1$ integrating over all the possible locations of the observation after the last motion u_{k-1},

$$p(c_k(i) = 1) = \int p(z_k(i) \mid c_k(i) = 1, x_k, M, O_k) \qquad (2)$$
$$p(x_k \mid x_{k-1}, u_{k-1}) p(u_{k-1}) du_{k-1}$$

We model the odometry u_{k-1} of the previous equation as a random variable with Gaussian distribution with a covariance matrix $P_{u_{k-1}}$. Consequently the location of the observation $z_k(i)$ is also a random variable given by the transformation of a point between two reference systems. Using the Jacobian of this transformation we propagate the uncertainty of the possible robot locations to the observations (see Fig. 2). The integration of Eq. 3 is implemented as a discrete weighted summation over all the possible locations of $z_k(i)$. For each possible robot pose x_k the value of $p(z_k(i) \mid c_k(i) = 1, x_k, M, O_k)$ is equal to:

$$p(z_k(i) \mid c_k(i) = 1, x_k, M, O_k) = \begin{cases} p_{static}, & \text{if } z_k(i) \in M \\ p_{dynamic}, & \text{if } z_k(i) \in O_k \\ 0.5, & \text{otherwise} \end{cases}$$

Those locations corresponding to static obstacles have a high probability, p_{static}, of generating static observations while those occupied by moving objects have a lower one $p_{dynamic}$. Note that we are using only positive information and thus all the free space and unknown areas are initialized to the non informative probability of 0.5. If the resulting probability $p(c_k(i) = 1)$ is above a certain threshold, the observation is classified as static.

Figure 2 illustrates the previous procedure for three different observations. For instance, the classification of observation two will depend on the support provided by the static and dynamic parts. Note that observations corresponding to new areas or new moving objects will usually have a $p(c(i))$ equal to 0.5. For these observations we now use negative information to classify as dynamic those observations surrounded by free space. The decision for those observations placed at the frontiers of known areas must be delayed until we get some more information in the next steps. These points are not

Algorithm 1 : Step k .

```
Iteration k
INPUT: $\hat{M}_{k-1}$,$\hat{x}_{k-1}$, $\hat{O}_{k-1}$, $z_k$, $u_k$.
Step 1: Prediction:
1.1- compute $\hat{x}_{k-} = p(\hat{x}_k \mid \hat{x}_{k-1}, u_{k-1})$
1.2- $\forall$ object $o^i$ compute $\hat{o}^i_{k-} = p(o^i_k \mid \hat{o}^i_{k-1})$
     $\hat{O}_{k-} = \{\hat{o}^1_{k-}, ...\hat{o}^n_{k-}\}$
Step 2: $x_k$ and $c_k$ estimation
2.1-$\hat{x}_k = \hat{x}_{k-}$
2.2 Maximize $p(z_k, c_k \mid M, x_k, O_k)$
repeat
   2.2.1- $\forall$ $z_k(i)$, compute $\hat{c}_k(i)$
   2.2.2- Compute $\hat{x}_k = argmax_x p(z_k|\hat{x_k}, \hat{M}_{k-1}, \hat{O}_{k-}, \hat{c}_k)$
until convergence
Step 3: Update:
3.1- update the map $\hat{M}_k$
3.2- Associate dynamic observations to each object $z_{o^i}$
3.3- $\forall$ object $o^i$ compute $\hat{o}^i_k = p(z_{o^i}|o^i)p(o^i \mid \hat{o}^i_{k-1}, \hat{x}_k)$
     $\hat{O}_k = \{\hat{o}^1_k, ...\hat{o}^n_k\}$
OUTPUT: $\hat{M}_k$, $\hat{x}_k$, $\hat{O}_k$
```

included in the optimization process but will be used in next steps to identify new static parts of the environment.

Once the classification of points and the position of the robot have converged, we proceed to update the grid map with those points classified as static. The grid map is updated using a technique similar to the one presented in [5] using the final pose estimate $\hat{x}_k$. As mentioned before, the map has a limited size which implies that we do not need to close big loops. However, one could apply techniques for global mapping([18], [10]) if necessary. Algorithm 1 illustrates the algorithm for step k.

B. Tracking the moving objects

In our current implementation we represent the moving objects as a set of independent extended Kalman filters, one per object. The state vector of each Kalman filter contains the estimated location and velocities of the object $o^i_k = \{x^i, y^i, \dot{x}^i, \dot{y}^i\}$. Given our particular application we model the motion of the moving objects with a constant velocity model with acceleration noise. Although moving people are highly unpredictable, we found that this model is enough to track them in most of the situations.

Once the measurements have been classified and the pose of the robot estimated, we proceed to update the filters using the dynamic measurements. The data association is done using the nearest neighbor rule. Besides the classical prediction and update steps for each filter, our system also has to perform some other tasks to correctly track the objects. First, it creates and removes filters as objects appear and disappear from the field of view of the sensor. It also has to perform merging and splitting operations to avoid several filters to track the same object or a filter to track several objects.

V. Using the Model Information for Tactical Planning and Obstacle Avoidance

The model builder presented in the previous section provides a map of the static parts of the environment and a set of filters tracking the moving objects. In this section we describe how this information is used by the rest of the modules (Figure 1). Further details about the architecture are described in [14].

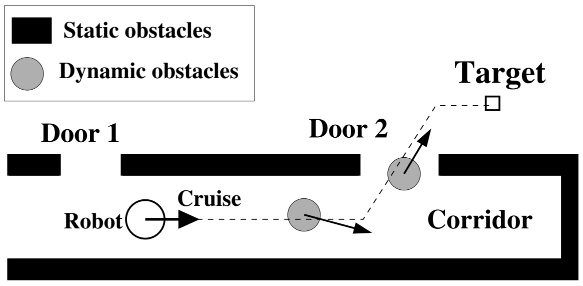


Fig. 3. Moving obstacles are not taken into account to compute the path.

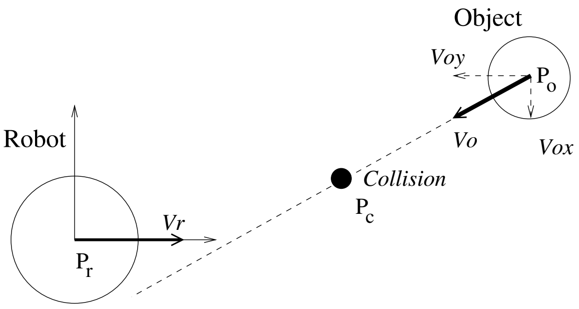


Fig. 4. Computation of the new object location for obstacle avoidance.

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[1] that move around (e.g. people). Thus, we use the static map as model for the planner. Figure 3 shows an example. The planner computes the cruise (main direction of the path) that points toward the end of the corridor. This is because the dynamic obstacles do not affect the planner. Notice that for a system that does not model the dynamic parts, *Door* 2 and the corridor would appear closed. Furthermore, in the example, if one of the dynamic obstacles stops it becomes an static object and would be integrated in the static map (blocking the path). Thus, they will be taken into account by the planner and the new cruise would point through *Door* 1 (avoiding the trap situation).

The obstacle avoidance module generates the collision-free motion to align the vehicle toward the cruise (computed by the planner). Here we use the map of static obstacles, since all the obstacles included in the map must be avoided. Furthermore, we use an approximation to the full obstacle avoidance problem with dynamic obstacles [6] which is described next. For each dynamic obstacle, we compute the time t_c at which the collision occurs in the direction of motion of the robot using the current vehicle and obstacle velocities ($\mathbf{v_r}$ and $\mathbf{v_o}$ respectively). We assume that both the object and the robot move with constant lineal velocities.

[1]A mobile object that stops (zero velocity) becomes a static object after a certain time period.

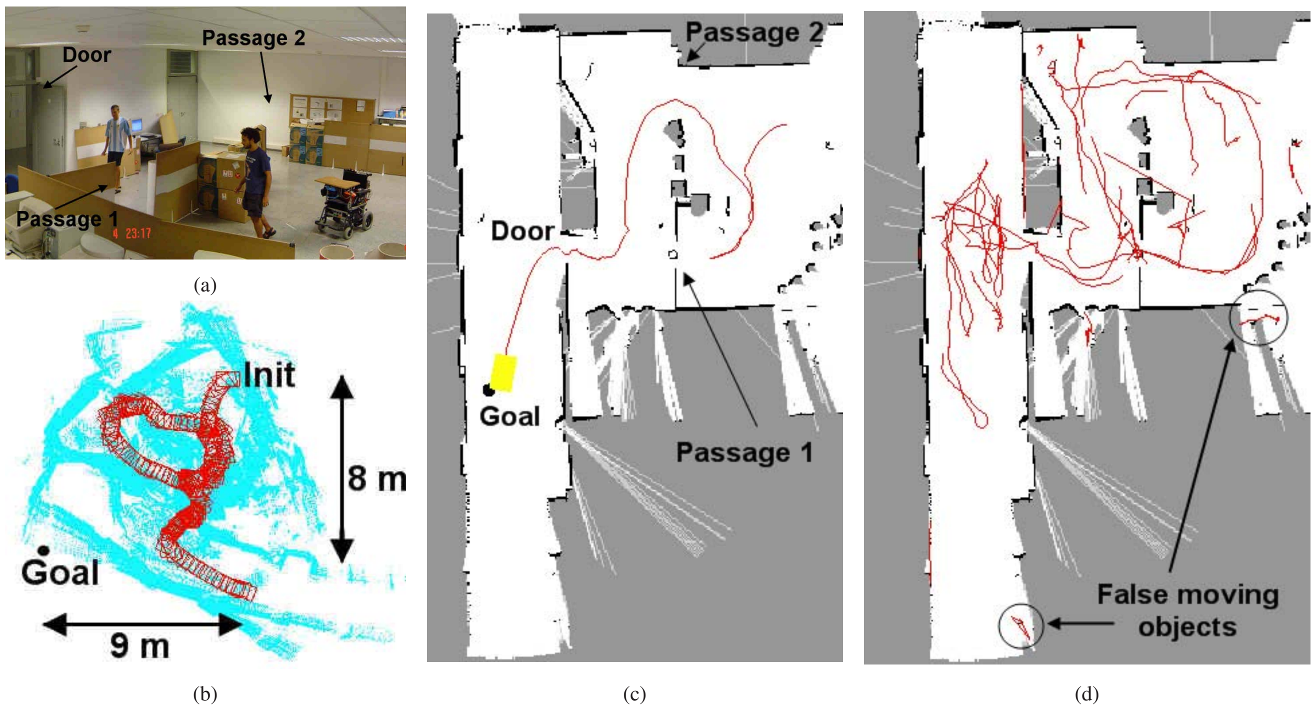


Fig. 5. These Figures show one experiment where the wheelchair drove out of the office. (a) A snapshot of the experiment. (b) real laser data and trajectory using the raw non corrected odometry. (c) the map built during the experiment and the vehicle trajectory. The map shows the occupancy probability 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.

$$t_c = \frac{p_{o_x} - (R_r + R_o)}{v_{r_x} - v_{o_x}} \tag{3}$$

where R_r is the radius of the robot, R_o the radius of the object and $\mathbf{p_o}$ is the current location of the obstacle. Then, we place the obstacle at the location $\mathbf{p_c}$ that will be in time t_c (Figure 4). The new location of the obstacle $\mathbf{p_c}$ is:

$$\mathbf{p_c} = (p_{o_x} + v_{o_x} t_c, p_{o_y} + v_{o_y}.t_c) \tag{4}$$

In fact, we want to avoid that the robot moves toward $\mathbf{p_c}$ since the collision will occur there. Let us remark that the new 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 ($t_c < 0$), 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 maneuver before than if the obstacle was considered static; and (ii) if there is no potential collision, the obstacle is not taken into account. This strategy heavily 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.

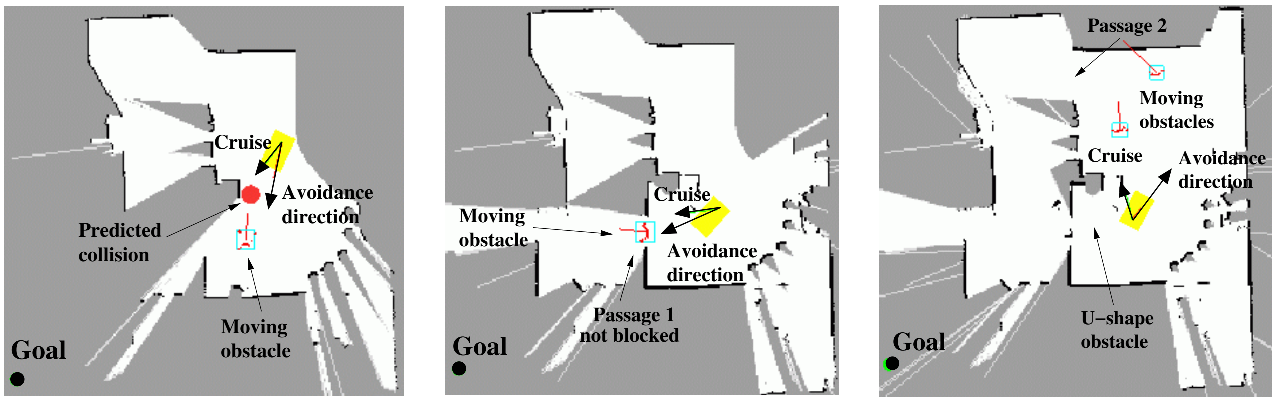
In summary, we have seen how the performance of the tactical planning and obstacle avoidance method are greatly improved by selectively using the static and dynamic information. Next, we describe the experimental results.

VI. Experimental results

For experimentation we used a commercial wheelchair that we have 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 $PentiumIII 850 Mhz$ at the frequency of the laser 5Hz.

We outline next one of the tests of the sensor-based navigation system in a difficult scenario due to the vehicle used and the nature of the surroundings. The wheelchair is a non holonomic robot with a rectangular geometry (it cannot move in any direction it sweeps an ample area when turns). The laser sensor is placed in the front part of the robot (0.72m) and has a $180°$ field of view. Thus, sometimes some of the static obstacles to avoid are out of the field of view and dynamic obstacles appear and disappear constantly. Furthermore, the ground is polished and the vehicle slides constantly with an adverse effect on the odometry. On the other hand the scenario is unknown and not prepared to move a wheelchair and there are many places where there is little room to maneuver. In addition, people turn the scenario in a dynamic and unpredictable place and sometimes modify the structure of the environment creating global trap situations.

The objective of the experiment was to drive the vehicle to a location that was out of the office (Figure 5a). Initially, the



(a) (b) (c)

Fig. 6. Three snapshots of the experiment. 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. (a) Moving obstacle going toward the robot, (b) a moving obstacle placed in the area of passage, (c) robot avoiding a trap situation.

vehicle proceeded toward the *Passage* 1 avoiding collisions with the humans that move around. Then, we blocked the passage creating a global trap situation that was detected by the system (Figure 5c). 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. We next outline some conclusions regarding the modeling module and the performance of the sensor-based system.

A. Modeling module results

The laser data collected during the experiment and the vehicle trajectory as given by the odometry are illustrated in Figure 5b. Here we devise that the vehicle odometry is very bad and it seems not possible to use it without a processing step. The size of the map is $20m \times 20$m ($5cm$ resolution grid). The map is centered around the current robot location, always contain the goal location and its resolution its enough for planning and obstacle avoidance. Figure 5c depicts the map of the static parts and the vehicle trajectory computed by the module. The map also includes some obstacles that stopped after moving for a certain period. The model can be used for tactical planning purposes and as short-time memory for obstacle avoidance.

The trajectories of the people tracked during the experiment are shown in Figure 5d. Most of them correspond to the people walking in the free space of the office. All the moving objects were detected by the modeling module. There were also some false positives due to misclassifications. They occurred mainly in two situations: when the laser beams were almost parallel to the reflected surface or when the beams missed an object because its height was similar to the height of the laser scan. Figure 5d depicts two of them which were included in the map when the system realized they were static.

Finally, Figure 6 illustrates different moments of the experiment. The rectangles represent the estimated location of the moving objects being tracked and contain the observations associated to each of them. The estimated velocity vector is represented by a straight line. The circle of Figure 6a represents the predicted collision according to the current velocities of the object and the robot. In the other figures the objects do not interfere with the vehicle trajectory.

B. Sensor based navigation system performance

There are two issues regarding the navigation performance. The improvement of the individual module behavior due to the selective usage of the static and dynamic information, and the improvement of the whole behavior of the system due to the integration architecture.

The planner computed at any moment the tactical information needed to guide the vehicle out of the trap situations (the cruise) but only using the static information. The most representative situations happened in the *Passage* 1. While the vehicle was heading this passage, people was crossing it. However, since the humans were tracked and labeled dynamic they were not used by the planner and thus the cruise points toward this passage (Figure 6b). Then, the vehicle aligned with this direction. Notice that systems that do not model the dynamic obstacle would consider the human static and the vehicle trapped within a U-shape obstacle.

Next, while the vehicle was about reaching the passage, a human placed an obstacle in the way. The vehicle was trapped in a large U-shape obstacle (Figure 6c). Rapidly, the object was identified static and included in the static map. Then, the planner computed a cruise that pointed toward the *Passage* 2. The vehicle was driven toward this passage avoiding the trap situation.

The obstacle avoidance module computed the motion taking into account the geometric, kinematic and dynamic constraints of the vehicle [12]. The method used the static information included in the map, but also the predicted locations of the objects computed using the obstacle velocities. Figure 6a depicts an object moving toward the robot, and how the predicted collision creates an avoidance maneuver before than

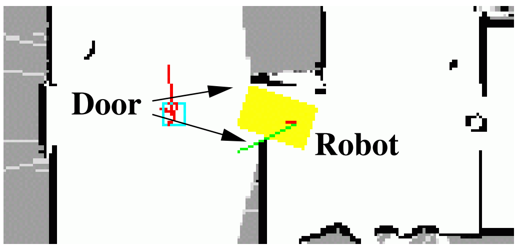


Fig. 7. Detail of the robot maneuver to cross the *Door*.

it would be obtained if the obstacle is consider static (case of the navigation systems that do not model the dynamic). Furthermore, obstacles that move further away from the robot are not considered. In Figure 6c the two dynamic obstacles were not included in the avoidance step (systems that do not model the dynamic would consider them).

The performance of obstacle avoidance module was determinant in some circumstances, specially when the vehicle was driven among very narrow zones. For example, when it crossed the door (Figure 7), there were less than 10cm at both sides of the robot. In addition, during the passage, the obstacle avoidance method computed motion between very near obstacles, and this movement was free of oscillations, and at the same time was directed toward zones with great density of obstacles or far away form the final position. That is, the method achieves robust navigation in difficult and realistic scenarios avoiding the technical limitations of many other existing techniques.

In summary, the modeling module is 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 an hybrid sensor-based navigation system that performs in difficult scenarios avoiding typical problems as the trap situations.

VII. Acknowledgments

This work has been partialy supported by the spanish projects MCYT-DPI2003-7986 and DGA-2004T04.

VIII. Conclusions

In this paper we have addressed two issues of great relevance in sensor-based navigation: how to model the static and dynamic parts of the scenario and how to make use of this information within a real sensor-based navigation system.

Our contribution in the modeling aspect is to formulate the DATMO and SLAM in such a way that 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 improving the resulting model.

The second issue is the integration of the modeling module in a real system taking advantage of the dynamic and static information. Our sensor-based navigation uses selectively 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 navigation schemes. The experimental results confirm that the system is able to drive the vehicle in difficult, unknown and dynamic scenarios.

References

[1] R. Arkin. Towards the unification of navigational planning and reactive control. In *Working Notes of the AIII Spring Symposium on Robot Navigation*, pages 1–6, Stanford University, 1989.

[2] K. Arras, J. Persson, N. Tomatis, and R. Siegwart. Real-time Obstacle Avoidance for Polygonal Robots with a Reduced Dynamic Window. In *IEEE Int. Conf. on Robotics and Automation*, pages 3050–3055, Washington, USA, 2002.

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

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

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

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

[7] D. Hahnel, 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.

[8] 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.

[9] H. Koyasu, J. Miura, and Y. Shirai. Recognizing moving obstacles for robot navigation using real-time omnidirectional stereo vision. *Int. Journal of Robotics and Mechatronics*, 14:to appear, 2002.

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

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

[12] J. Minguez and L. Montano. Robot navigation in very dense and cluttered indoor/outdoor environments. In *Int.Federation of Automatic Control IFAC 15th World Congress*, Barcelona, Spain, 2002.

[13] J. Minguez and L. Montano. Nearness diagram navigation (nd): Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1), 2004.

[14] J. Minguez, L. Montesano, and L. Montano. An architecture for sensor-based navigation in realistic dynamic and troublesome scenarios. In *IEEE Int. Conf. on Intelligent Robot and Systems*, Sendai, Japan, 2004.

[15] E. Prassler, J. Scholz, and P. Fiorini. Navigating a robotic wheelchair in a railway station during rush hour. *International Journal of Robotics Research*, 18:711–727, 1999.

[16] 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.

[17] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[18] 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.