Integration of Planning and Reactive Obstacle Avoidance in Autonomous Sensor-Based Navigation

Javier Minguez
Instituto de Investigación en Ingeniería de Aragón
Dept. Informática e Ingeniería de Sistemas Universidad de Zaragoza, Spain
jminguez@unizar.es

***Abstract*— This paper presents a sensor-based navigation system to safely drive vehicles in realistic scenarios. Three modules with the following functionalities compose the system: model builder, tactical planning and obstacle avoidance. These modules are integrated within a planner - reactor architecture that supervises and coordinates them in order to carry out the motion task. The emphasis of the paper is in the planning aspect and in its integration with the obstacle avoidance and modeling module. The advantage of this navigation system is to achieve a robust and trustworthy navigation in difficult scenarios, which remain troublesome for many of the existing systems. In order to validate the system, we present experiments with a wheelchair vehicle transporting a human among locations in an office type scenario.**

I. Introduction

The general task of an autonomous motion system is to generate movement free of collisions between successive locations. Their design usually involves the model construction, the deliberative planning and the obstacle avoidance. The model builder constructs a representation that is the base for the deliberation and local memory the avoidance behavior, the planner module generates global plans and the obstacle avoidance computes the local motion. The sensor-based systems made up as synthesis of these functionalities mainly differ in the integration between the planner and the reactor (i.e. how the obstacle avoidance uses the information available of the planner), and in the tools used to implement each module. We present next related work following these two premises.

One way to specify the interaction between deliberation and reaction is to consider planning as a component that fixes the composition between different motion behaviors during execution [2]. Another possibility is to use the planning to advise the reactive control [1], or as a system that adapts parameters of the reactive component based on the evolution of the surroundings [15]. In both cases, planning plays a tactical role while the reactor has the execution degree of freedom. In sensor-based motion a common strategy is to compute a path and use its course to direct the reactive module [7], [18], [3] (we follow here this strategy). Other techniques compute a path that is deformed in execution based on the evolution of the environment (in the workspace [8] or in the configuration space [21]). Alternatively [27] presents a strategy to create trees of paths obtained by executing the reactive algorithm some steps ahead of the execution. Another possibility is to compute a channel of free space that contains sets of ways, leaving the choice up to the execution [9].

Closely bound with these issues are the choice and implementation techniques for each module. With regard to the construction of a model, in indoor environments, the occupancy grids are usually used with ultrasounds [10], [5], [23] and with laser [7], [18], [3]. With regard to the planners, these systems use efficient numerical techniques on grids that are executed in real time [4], [25]. Another key issue is the obstacle avoidance method, where some systems use the potential field methods [13], those based on intermediate sets of commands [6], [24], [12], or those based on high-level information [17], [20].

In this paper we present an autonomous navigation system composed by these three subsystems and a architecture of integration [19]. The emphasis of the paper is in the planning aspect and in its integration with the obstacle avoidance system. On one hand, we use a planner that replanns only in the areas that have changed from the previous sensory perception, and that affect the computation of the path [22]. This results in a very efficient strategy with regard to previous works that intensively explore the space at each cycle irrespective of the environmental changes (in these methods the size of the model and difficulty of the scenario are always bounded due to the compromise with the real-time requirement).

Secondly, the planner was integrated with the model and the obstacle avoidance method [17] in a planner - reactor architecture [15] to build a complete autonomous navigation system. This synergy results in a very efficient way to address planning and reaction from both, robustness and computational point of view. Although the design of these systems is not new, what remains still inaccessible for many of the above-mentioned techniques is to carry out a robust, efficient and trustworthy navigation when the environments are very complicated. The system displayed here avoids many of the limitations of related works, robustly navigating in these problematic scenarios. To validate the system we used a commercial wheelchair equipped with two on-board computers and with a planar laser. We report experiments in a realistic office type scenario.

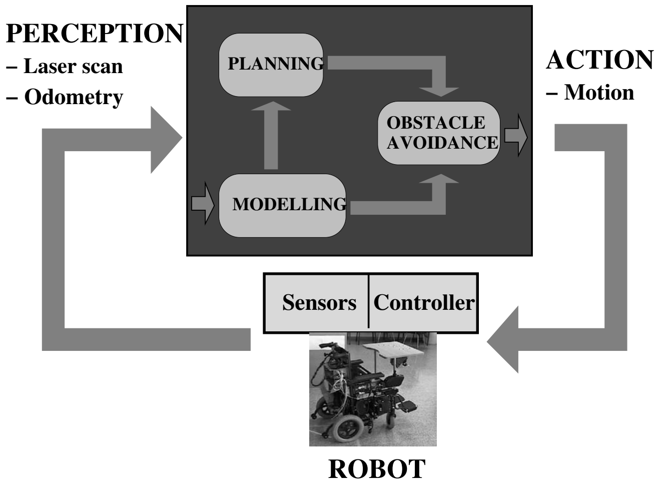


Fig. 1. Overview of the sensor-based navigation system

II. Overview of the System

We give in this Section a global vision of the sensor-based system, which is formed by an architecture that integrates three modules with the following functionalities: model construction, motion planning and obstacle avoidance:

- **Model Builder Module**: construction of a model of the environment (to increase the spatial domain of the planning and used as local memory for obstacle avoidance). We use a binary occupancy grid that is updated whenever a new sensory measurement is available. Furthermore, we employ a scan matching technique to improve the vehicle odometry before integrating any new measurement in the grid.
- **Planner Module**: extraction of the connectivity of the free space (used to avoid the cyclical motions and trap situations). The idea behind the planner is to focus the search locally in the areas where the changes in the scenario structure have occurred and affect the computation of the path. The planner avoids the local minima and is computationally very efficient for real time implementations.
- **Obstacle Avoidance Module**: computation of the collision-free motion. We chose the Nearness Diagram Navigation (ND method in short), which is based on selecting at every moment a navigational situation and to apply a motion law adapted for each one. This method has demonstrated to be very efficient and robust in environments with little space to maneuver.

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 information of the changes in obstacle and free space in the model is used by the planner module to compute the course to follow to reach the goal. Finally, the avoidance module uses the information of the obstacles contained in the grid and information of this tactical planner to generate the motion (to drive the vehicle free of collisions towards the goal). The vehicle controller executes the motion 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 design of the modules and the integration architecture with emphasis in the planning module.

III. Module Design and Integration

We describe in this Section the model builder module (Subsection III-A), the planner module (Subsection III-B), the obstacle avoidance method (Subsection III-C) and the architecture of integration (Subsection III-D).

A. Model Builder Module

The function of this module is to integrate the sensorial measurements to construct a model of the environment (in our case a local map). We chose a binary occupancy grid because is an efficient structure to use from which it turns out simple to represent the obstacle and the free space. The cells are *occupied* and *free* since the laser used has a high precision in indoor environments. The grid has a fixed size that represents a limited part of the workspace (large enough to represent the portion of space necessary to solve the navigation, and to have the obstacles required to avoid collisions are always around the robot).

The design of this module includes two parts: (i) To improve the vehicle odometry, we use a scan matching technique with the information provided by the laser [14]. This technique searches for correspondences between two consecutive laser scans in order to estimate the rigid motion. (ii) To integrate a scan in the model, obstacle points are marked occupied, an all the cells over the lines that go from the sensor position to the obstacle points are marked as free. We implemented this procedure using the *Bresenham* algorithm [11].

One advantage of this model is that the odometry is improved via a scan matching technique. Although, these techniques do not guarantee global consistency, its precision is enough to build the local map needed by the other modules. This will be important latter for vehicles with poor odometry (as is the case here) since the model will be used for planning purposes and memory for obstacle avoidance. There are also other issues to highlight: (i) the last laser scan integrated in the grid does not have odometry errors with respect to the present position. Only the cells not updated with this scan accumulate odometry errors, which are, however, mitigated by the scan matching technique. (ii) The grid reflects the change in dynamic environments rapidly updating all the area covered by the last scan, and (iii) the spurious measurements are eliminated from the grid as new measurements are added. For all these reasons we think that this model is well suited as representation for the planning and local memory for the reactive module in unknown and dynamic scenarios.

B. Planning Module

This module uses the D*Lite planner [22] to obtain tactical information to avoid the trap situations and the

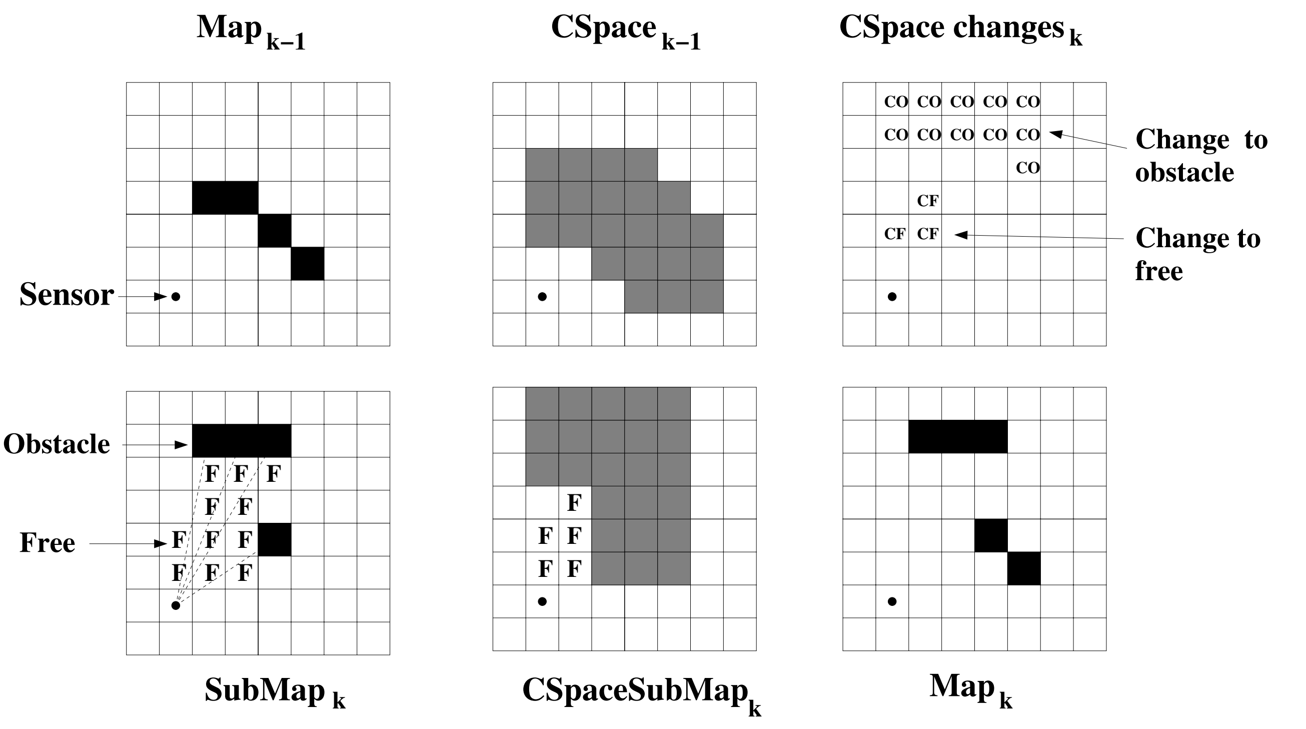


Fig. 2. This Figure displays how the grid model is updated with the last perception and how the changes are computed in configuration space. First, we compute Submap$_k$ using the new laser scan. Then, we compute the configuration space of both, the previous map (Cspace$_{k-1}$) and the new submap (CSpaceSubmap$_k$). (We assume that the robot is a blob of eight cells.) Finally, we merge the maps to obtain the new map Map$_k$ (input of the obstacle avoidance), and we compute the changes in the configuration space Cspace changes $_k$ (input of the planner).

cyclic motions. 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 first part of this module computes changes in the configuration space (cspace) from free space to occupied and vice versa given the current perception. We assume that the vehicle is circular and thus the cspace is represented by $\mathbb{R}^2$. The obstacle representation is obtained by enlarging each obstacle with the radius R of the circle inscribed to the robot.

Let Map$_{k-1}$ be the grid at the previous step $k-1$ and Cspace$_{k-1}$ the cspace computed from this grid map. Let SubMap$_k$ be the grid computed only using the last laser measurement, and CspaceSubMap$_k$ the corresponding cspace. We compute the current map of changes in configuration space CspaceChanges$_k$ as:

CspaceSubmap$_k^{i,j}$	Cspace$_{k-1}^{i,j}$	CspaceChanges$_k^{i,j}$
Occupied	Free	Change to Obstacle
Free	Occupied	Change to Free

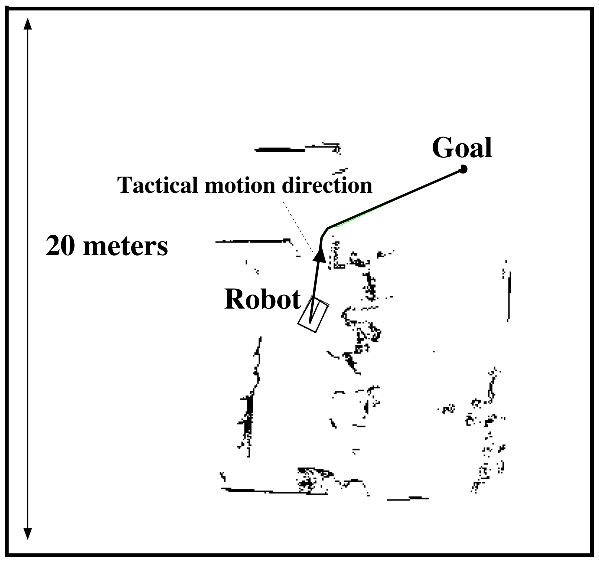
The result is the model of changes in configuration space CspaceChanges$_k$ (Figure 2). This is the input of the D* Lite planner.

The second part of this module computes a path in configuration space from the current vehicle location towards the goal. The D* Lite planner models the navigation problem with a graph. A vertex represents a location of the vehicle, and has associated an edge that points to the adjacent vertex to reach the goal. From any vertex, the shortest path to the goal can be computed by following the edges. Other additional edges codify the cost of going from one vertex to the adjacent. Changes in the environment modify these costs. Then, the planner locally explores and rearranges the edges affected by the changes and that are relevant to compute the shortest path.

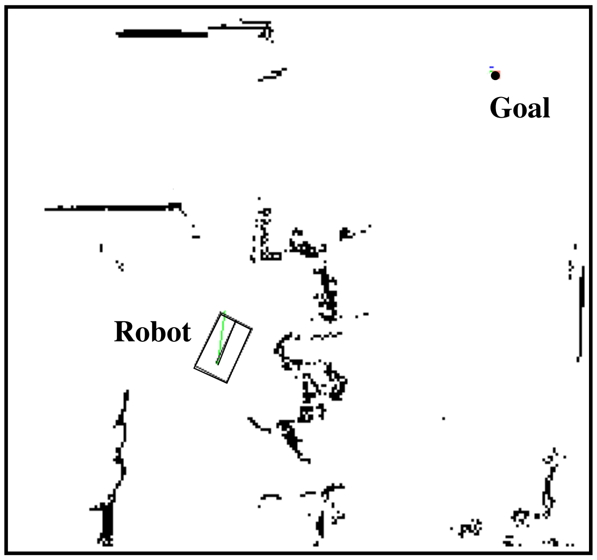
In our problem, we model the graph as a grid where the vertices are the cells, and the edges point towards one of the adjacent cells (eighth connected). From each cell, the cost to reach an adjacent free cell is one and infinity for an obstacle one.

Initially the graph is computed starting from the goal (similar to an A* strategy). Then, one can compute the shortest path from any cell by following the edges. In execution time there are changes in the grid and thus in the configuration space (this is the information computed in the previous step). The cost edges are modified using these changes, and propagated in the graph. Finally, the path is computed from the current vertex with an steepest descendent strategy.

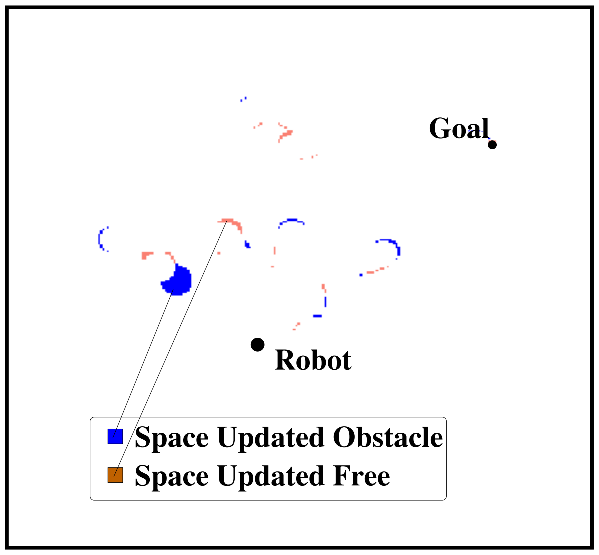
The Figure 3 shows how the planner computes the path at time k . Figure 3a,b show an overview of the grid and a zoom around the robot. Using the last perception we first



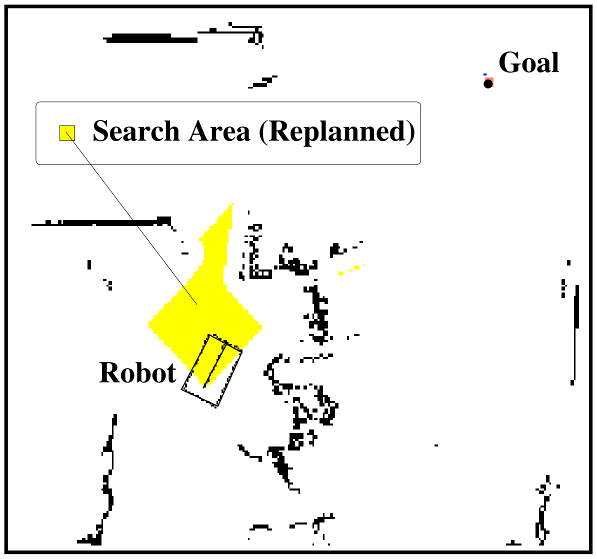
(a)



(b)



(c)



(d)

Fig. 3. These Figures show the usage of the planner in a real experiment. (a) The robot and current model. (b) Zoom of (a) around the robot. (c) Changes in configuration space due to the last perception. (d) Cells explored to recompute the shortest path [displayed in (a)].

compute the changes in the configuration space (Figure 3c). This is the input to the planner. The changes are propagated over the relevant cells to compute the shortest path from the current vehicle location (the cells explored are displayed in Figure 3d). Finally the path is computed by following the edges (Figure 3a). The advantage is that this planner does not recompute from scratch the path in each iteration. It locally propagates the changes. Notice how this strategy is by far more efficient than exploring the complete grid to compute the path (up to two orders of magnitude [26]).

From the path we obtain two types of information: first, if it is possible to reach the goal from the present position (if the path exists the planner always find it since it is free of local minima). Secondly the *tactical motion direction* (main course of the first part of the path). Notice that this direction will not be used to direct the vehicle (since this degree of freedom will be handled by the following module), but as the main course of the motion.

C. Obstacle Avoidance Module

The ND+ method [20] is an improvement of the ND method [17]. The design is based on defining a set of situations to represent the navigational problem, and how to act in each of them (actions). In real time, at each control cycle a situation is selected and the corresponding action is executed computing the motion.

The advantage of this method is that it employs a divide and conquer strategy based on situations to simplify the difficulty of navigation. Thus this technique is able to deal with more complex navigation cases than other methods (usually these cases arise in environments where there is little space to maneuver like for example a narrow door).

In particular, the ND+ method avoids most of the problems that other techniques present in these circumstances, like the local trap situations, the oscillating movements, or the impossibility to move towards certain zones with high obstacle density or far away from the goal direction (see [17] for a discussion on this topic). As it will be illustrated in the experimental results, these properties were determinant to navigate in the majority of realistic environments. The ND+ method improves the previous ND method with new navigational situations and a new design of the motion laws (to have motion continuity in the most common transitions between situations). Another advantage of the ND+ method is that is very efficient and it can be used when required without imposing a significant time penalty [20].

D. Integration Architecture

The architecture integrates the modules considering the limitations and restrictions imposed by the mechanical (sensors and actuators) and logical parts (computers) of the robot [19]. The architecture has a synchronous planner - reactor configuration, where both parts use the model constructed in execution time. The functionality of the modules is:

- The model construction, used for the planner to compute the changes in configuration space and for the obstacle avoidance method as local memory.
- The planner computes the tactical motion direction (to guide the obstacle avoidance method).
- The obstacle avoidance method computes the motion commands to avoid collisions (with the obstacles represented in the model) while following the motion direction provided by the planner (that codifies the main cruise to reach the target).

This hybrid architecture allows to concentrate the best of worlds both (deliberative and reactive), since the information of the planning allows to guide the motion towards zones in which trap situations do not take place, and the reactive component directs the execution with fast reactions to the evolution of the environment (considering in addition non visible zones from the present position available in the model). All the modules have been integrated in such a way that the control loop is always closed at $5Hz$ (sensor frequency) with a motion command available (there are no dead states).

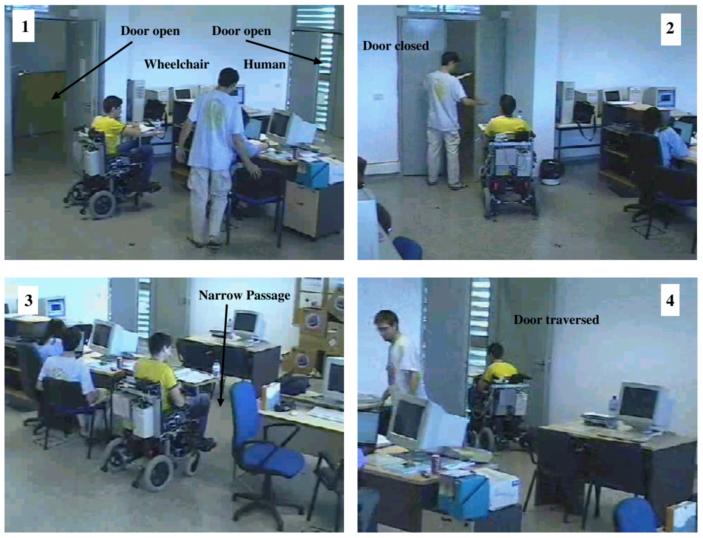
IV. Experimental Results

For experimentation, we used a commercial wheelchair that we have equipped with a SICK laser and with two on-board computers (two $PentiumIII850Mhz$, one of them is used for motion control purposes and in the other one the computations associated to the architecture were carried out). The vehicle is rectangular $(1.2 \times 0.7 meters)$ with two driving wheels that work in differential-driven mode. We set the maximum operational velocities to $(v_{max}, w_{max}) = (0.3\frac{m}{sec}, 0.5\frac{rd}{sec})$ due to the application context (human transportation). Our model is a grid that represents $20m \times 20m$ (400×400 cells and $0.05m$ each cell). The portion of the environment represented in the model is sufficiently large to include the target (where we have to drive the robot), and the resolution is enough for navigation purposes.

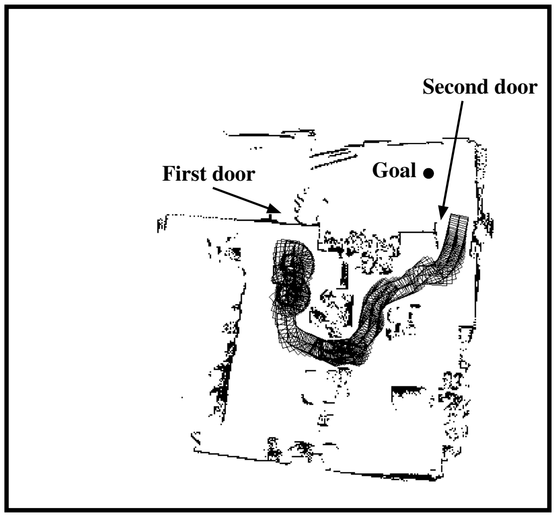
The experiments outlined here are particularly difficult due to the vehicle used, the type of task and to the nature of the surroundings. The wheelchair is a non-holonomic robot with the driving wheels in the back part, thus it cannot move in any direction and sweeps an ample area when it turns. In addition, since the vehicle transports humans, a smooth trajectory is desirable, avoiding shaking behaviors (i.e. the vehicle has geometric, kinematic and dynamic constraints). The laser sensor is placed in the front part of the robot (0.72m) and has a $180°$ field of view, thus some obstacles to avoid are not visible from the present position. Furthermore, the ground was just polished and the vehicle slides constantly with an adverse effect on the odometry. On the other hand the surroundings are unknown, since there are elements in the office like chairs, tables whose position cannot be established a priori (although the walls could be known, unfortunately they are not visible by the sensor since the furniture hides them). This scenario is not prepared to move a wheelchair and in many places there is little room to move. In addition, people working in the office turn the scenario in a dynamic and unpredictable place, and as well, sometimes the structure is modified creating global trap situations.

In the experiment the wheelchair had to drive the human until a position outside the office. First, the vehicle moved towards the closest visible door (snapshot 1 of Figure 4a). During the motion, a person closed the right leaf of the door so that the wheelchair did not fit (the vehicle was trapped in a large U-shape obstacle). Rapidly, the vehicle modified its way returning backwards (snapshot 2) in order to find the exit. Next the robot traveled avoiding collisions with the furniture and a person who moved bothering the normal progression of the vehicle (snapshot 3). In the center of the office, the robot detected a half open door but sufficiently wide to fit in. Then, the vehicle proceed to the door and maneuvered until crossing it, leaving the office and reaching the target position (snapshot 4). Next we discuss some technical aspects of the experiment.

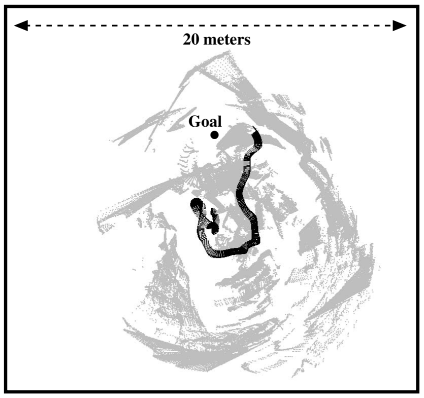
The data obtained from the laser and the odometry during the experiment are shown in Figure 4c. The main conclusion is that the odometry is quite bad and hardly could be used to deliberate or to compute collision free motion. Nevertheless, the modeling module manages this information and constructs a reasonable model (Figure 4b). This is because the scan matching technique improves the odometry so that the information is properly integrated in the grid. Furthermore, since the odometry is locally ameliorated, the vehicle approaches nearer to the destination. Another advantage is that the model represents rapidly the change (the surroundings are not known and dynamic). This is because the complete area swept by the last scan is updated in the grid (the occupied and free space are updated), so that the new obstacle information is included and those obstacles that currently are not present are eliminated. The planning and obstacle avoidance modules share



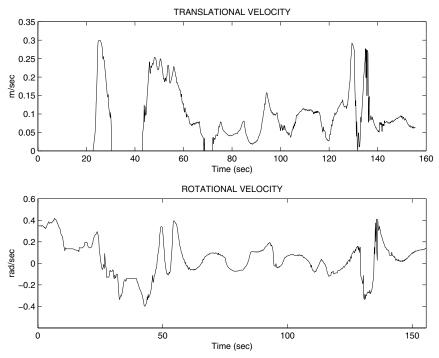
(a)



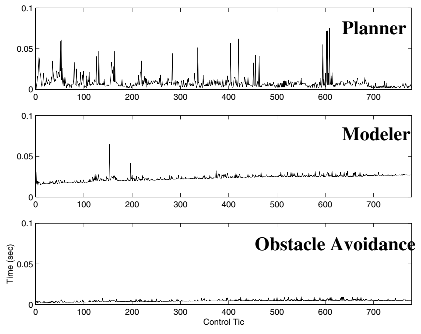
(b)



(c)



(d)



(e)

Fig. 4. These Figures show one experiment where the wheelchair drove a human out of an office. (a) Some snapshots of the experiment, (b) the model built during the experiment and the vehicle trajectory, (c) real laser data and trajectory using the odometry, (d) motion commands and (e) computation time of each module.

the benefits of this model. The changes in the scenario are rapidly computed and used by the planner to compute courses to follow (to avoid traps), and the information of the non visible obstacles from the present position is always available for collision avoidance purposes (this case is understood when the door was crossed, snapshot 4 of Figure 4a), since once the sensor has passed the door the frame is not detected).

The planner computed at any moment the tactical information needed to guide the vehicle out of the trap situations. The most representative situation happened when the door was open and suddenly was closed next (snapshots 1 and 2 of Figures 4a). The course of the planner rapidly changed pointing backwards (that directed the motion outside of the end zone). The navigation system avoids the trap situations and the cyclic behaviors by using the information of the planner in each iteration.

The obstacle avoidance module computed the collision free motion during the complete experiment taking into account the geometric, kinematic and dynamic constraints of the vehicle [16]. The performance of this module was determinant in some circumstances, specially when the vehicle was driven among very narrow zones [like for example when it crossed the door (snapshot 4 of Figure 4a)]. In addition, during the run, the ND+ method computed motion between very near obstacles, and this movement was free of oscillations and irregular behaviors (see the velocity profiles in the Figure 4d and the vehicle path in Figure 4b), and at the same time was directed towards zones with great density of obstacles or far away form the final position (any direction of movement can be obtained). That is, the method achieves robust navigation in difficult and realistic scenarios avoiding the technical limitations of many other existing techniques.

The final issue is the computation time of each module depicted in Figure 4e. The average execution time is 0.025sec for the model module, 0.02sec for the planner and lower than 0.005sec for the reactive method. It is important to remark the efficiency of the planner that works in a 400×400 grid (since it only explores the space based on the environmental changes, and does not replan at each iteration from scratch). With this rates all the modules worked synchronously within the cycle of the sensor 0.2sec. Furthermore the CPU is free the majority of the time to be used by some upper levels that could require it.

To conclude, this experiment illustrates how the system proposed here generates robust and trustworthy navigation in unknown, dynamic and difficult scenarios. That is, to move vehicles in realistic environments where the things are not where one likes, people move around, there is little site to maneuver and the well-known trap situations are usual.

V. Conclusion

We have presented in this paper a sensor-based navigation system that is made up of three modules: a model constructor, a planning method and an obstacle avoidance method. Although some of these techniques derive from already existing works, the main contribution here is the planner used and its integration with all the modules in the system. Furthermore, it is important to stress that the design of sensor-based systems is not new. What remains still inaccessible for the many navigation systems is to carry out a robust and trustworthy navigation when the environments are very complicated. This is the advantage of the work displayed here.

VI. Acknowledgement

We want to thank Luis Montesano for participating in this project with technical discussions and contributing with many ideas. This project was partially supported by Spanish *Ministerio de Educación y Ciencia* under the project MCYT DPI2003-07986.

References

[1] P. Agree and D. Chapman. What are the plans for. *Journal for Robotics and Autonomous Systems*, 6:17–34, 1990.

[2] R. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *IEEE International Conference on Robotics and Automation*, pages 264–271, Raleigh, USA, 1987.

[3] 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.

[4] J. Barraquand and J. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *Intelligent Symposium on Intelligent Control*, pages 340–346, Albany, 1989.

[5] J. Borenstein and Y. Koren. Histogramic in-Motion Mapping for Mobile Robot Obstacle Avoidance. *IEEE Journal on Robotics and Automation*, 7(4):535–539, 1991.

[6] J. Borenstein and Y. Koren. The Vector Field Histogram–Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7:278–288, 1991.

[7] 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.

[8] O. Brock and O. Khatib. Real-Time Replanning in High-Dimensional Configuration Spaces using Sets of Homotopic Paths. In *IEEE Int. Conf. on Robotics and Automation*, pages 550–555, San Francisco, USA, 2000.

[9] W. Choi and J. Latombe. A reactive architecture for planning and executing robot motion with incomplete knowledge. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 24–29, Osaka, Japon, 1991.

[10] A. Elfes. Sonar-based Real-world Mapping and Navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[11] J. Foley, A. V. Dam, S. Feiner, and J. Hughes. *Computer Graphics, principles and practice*. Addison Wesley edition 2nd, 1990.

[12] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997.

[13] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. Journal of Robotics Research*, 5:90–98, 1986.

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

[15] D. Lyons and A. Hendriks. Planning, reactive. In S. Saphiro and J. Wiley, editors, *Encyclopedia of Artificial Intelligence*, pages 1171–1182. 1992.

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

[17] 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.

[18] J. Minguez, L. Montano, N. Simeon, and R. Alami. Global Nearness Diagram Navigation (GND). In *IEEE International Conf. on Robotics and Automation*, pages 33–39, Seoul, Korea, 2001.

[19] 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.

[20] J. Minguez, J. Osuna, and L. Montano. A Divide and Conquer Strategy to Achieve Reactive Collision Avoidance in Troublesome Scenarios. In *IEEE International Conference on Robotics and Automation*, Minessota, USA, 2004.

[21] S. Quinlan and O. Khatib. Elastic Bands: Connecting Path Planning and Control. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 802–807, Atlanta, USA, 1993.

[22] A. Ranganathan and S. Koenig. A reactive architecture with planning on demand. In *International Conference on Robotics and Automation*, page 14621468, Las Vegas, Nevada, 2003.

[23] 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.

[24] R. Simmons. The Curvature-Velocity Method for Local Obstacle Avoidance. In *IEEE Int. Conf. on Robotics and Automation*, pages 3375–3382, Minneapolis, USA, 1996.

[25] A. Stenz. The focussed d^* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, Montreal, CA, 1995.

[26] A. Stenz. The Focussed D^* Algorithm for Real-time Replanning. In *Proceedings of the Intenational Joint on Artifcial Intelligence*, pages 1652–1659, 1995.

[27] I. Ulrich and J. Borenstein. VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 1572–1577, 1998.