Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

An Architecture for Sensor-Based Navigation in Realistic Dynamic and Troublesome Scenarios

Javier Minguez Luis Montesano Luis Montano
Instituto de Investigación en Ingeniería de Aragón
Dept. Informática e Ingeniería de Sistemas Universidad de Zaragoza, Spain
Email: {jminguez,lmontesano,montano}@unizar.es

Abstract— **We address here a sensor-based navigation system to safely drive vehicles in realistic scenarios. The system is composed by three modules with the following functionalities: model builder, planning and reactive motion. These modules are integrated within a planner - reactor architecture that supervises and coordinates them in order to carry out the motion task. The advantage of our 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

Everyday, it increases the research and industrial interests of the robotics community in improving the autonomy degree of the systems. This is driving the development of robots that work within an ample rank of conditions, which are really useful when they operate during long periods of time without human intervention. One of the key aspects in this development is the mobility because this ability opens the possibility of adding a great variety of subsystems with other functionalities (increasing the degree of autonomy). The nature of the surroundings where the robot carries out the task is closely bound with the mobility aspect. For example, in applications like wheelchair robots, surveillance robots or service robots, the environment is not completely specifiable with an a priori map and can be dynamic. In these circumstances, it is clear the necessity of sensors to collect information of the environment in order to adapt the movement to any new contingency or event. The sensor-based navigation systems appear as a natural choice in these circumstances. Nevertheless, the difficulty of these techniques is to move vehicles in very complicated environments, that normally are those where there is very little room to maneuvre, are highly dynamic or create the well-known trap situations. We address here the navigation with a wheelchair vehicle under these work conditions.

Within the mobility of the vehicle, the sensor-based motion system is the part in charge of generating movement free of collisions between successive positions. The design of these systems is determined by diverse factors involved in this question, like the model construction, the deliberative planning and the motion generation. The model builder constructs a representation which is the base for the deliberation and which provides with memory the reactive behavior, the planner module generates global plans and the reactive module computes the local motion. The sensor-based systems made up as synthesis of modules with these functionalities mainly differ in the interaction between the planner and the reactor (i.e. how the reactive navigation 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.

To specify the interaction between deliberation and reaction, one possibility is to see the planning like a component that fixes the composition of different behaviors during the execution [2]. These behaviors are implemented with potential fields [14], so that modifying its weights modifies the global behavior of the system. Another possibility is to use the planning like advisor of the reactive control [1], or like a system that adapts parameters of the reactive component based on the evolution of the surroundings [17]. In both cases, the planner has a tactical role leaving the execution degree of freedom to the reactor. In this context, a common strategy is to compute a path and use its course to direct the reactive module [23], [20], [8]. Other methods compute a path, which is deformed according with the changes in scenario (the path is computed in the workspace [9] or in the configuration space [22]). Other techniques create trees of paths obtained a number of stages before the execution [26]. Another possibility is to compute a channel of free space that contains sets of ways, leaving to the execution the selection of one of them [10].

Closely bound with this issues is the choice and implementation of the techniques for each module. With regard to the construction of a model, in indoor environments, the occupancy grids are usually used with ultrasounds [11], [6], [23] and with laser [8], [20], [4]. With regard to the planners, these systems use efficient numerical techniques on grids that are executed in real time [5], [25]. Another key issue is the reactive method, where some existing systems use the potential field methods [14], [15], those based on sets intermediate of commands [7], [24], [13], or those based on high-level information [19], [21].

We describe here the design and verification of a sensor-based system made up of three modules whose synergy carries out the motion task. The model builder constructs and manages an occupancy grid and uses a scan-matching technique to improve the vehicle odometry [16]. The planner module is a new technique based on computing the existence of a *tunnel* of free space to reach the destination, and the reactive module is a technique that employs a "di-

0-7803-8463-6/04/$20.00 ©2004 IEEE

vide and conquer" strategy based on situations to simplify the difficulty of the navigation [21], [19]. The synthesis of these modules is carried out within a synchronous planner-reactor architecture [17].

Although the design of sensor-based systems is not new, what remains still inaccessible for the great majority of the above mentioned techniques is to carry out a robust and trustworthy navigation when the environments are very complicated. Our system differs from previous works in the choice and implementation of the particular techniques and in the architecture of integration. All these aspects together, compose a system that avoids 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 range-finder.

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 reactive navigation:

- **Model Builder Module**: construction of a model of the environment (to increase the spatial domain of the planning and used as local memory for the reactivity). We use a binary occupancy grid that is updated whenever a new sensory measurement is available. The grid has a limited size and travels centred with the robot. Furthermore, we employ a scan matching technique to improve the vehicle odometry before integrating any new measure in the grid. Although, a scan matching technique does not guarantee global consistency, its precision is enough to build the local map needed by the other modules.
- **Planner Module**: extraction of the connectivity of the free space (used to avoid the cyclical motions and trap situations). We have developed a new planner that computes the existence of a path that joins the robot and goal locations. 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.
- **Reactive Navigation 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 obstacles and free space in the grid is used by the planner module to compute the course to follow to reach the goal. Finally, the reactive

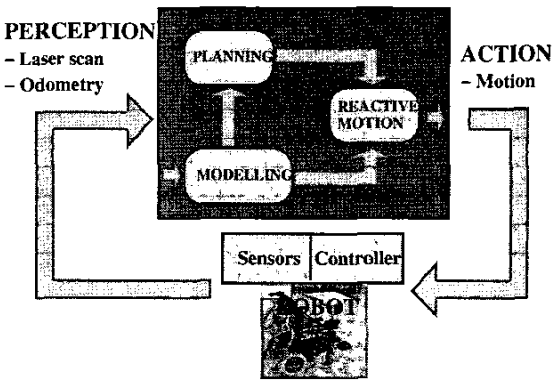


Fig. 1. Overview of the sensor-based navigation system

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 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 design of the modules and the integration architecture.

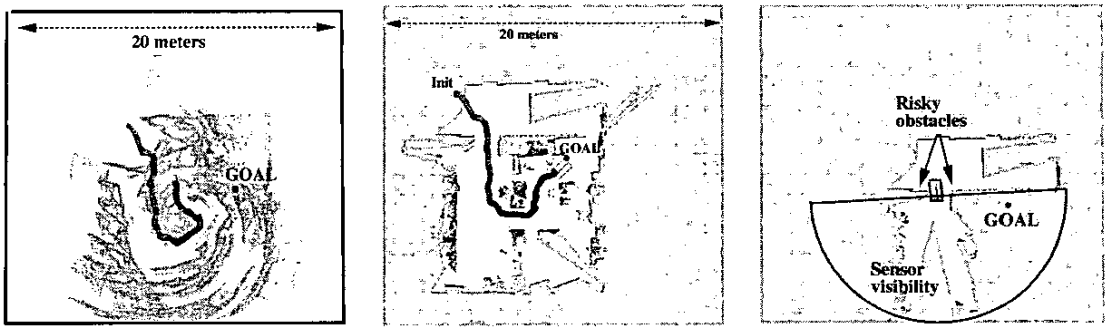
III. Module Design and Integration

We describe in this Section the model builder module (Subsection III-A), the planner module (Subsection III-B), the reactive 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 measures 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 obtain the free space (the one of interest for the movement). The cells are *occupied* and *free* since the laser used has a high precision in indoor environments (we do not use traversability factors). 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 whose position is recomputed to maintain the robot in its central zone (the obstacles required to avoid collisions are always around the robot). The design and supervision of this module include three parts: (i) the use of a technique of scan matching to improve the vehicle odometry. (ii) the integration of the laser measures in the model, and (iii) the supervision of model position to maintain the robot centred:

- To improve the odometry of the vehicle we use a scan matching technique, which refines the odometry readings using the information provided by the laser. We use the *Iterative Dual Correspondence* algorithm [16], which search for corresponces between two



(a) (b) (c)

Fig. 2. These Figures show the performance of the model in a real experiment. (a) The trajectory of the robot given by the odometry and the laser data. (b) The model constructed in execution and the trajectory of the robot provided by the scan matching technique. (c) Model available when the vehicle was crossing a door and the risky obstacles are not detected by the laser, but they are available in the grid.

consecutive laser scans in order to estimate the rigid motion. This algorithm does not require to extract any specific kind of features and consequently is well suited to unstructured environments.

- To integrate a scan in the model, the cells that corresponds to the position of the obstacle points are placed occupied, and all the cells over the lines that join the sensor position and the obstacle points are filled up free. We implemented this procedure using the *Bresenham* algorithm [12] which is optimal in the number of cells visited to project a line in a grid. The use of this algorithm considerably reduces the integration time of a sensorial measurement.
- To keep the robot located in the central zone of the grid, we define an area denominated *control zone*. When the robot leaves this zone, the new position of grid is recomputed to centre the robot within it. With this strategy the robot is always in the central zone of grid whose position does not change until the robot does not leave it. The recomputation of the grid position is always made in multiples of the cell's dimension and the rotation is not allowed (this strategy reduces the dissemination of false information of the obstacles in the cells, which is an important source of error). In addition, this strategy can be efficiently implemented with displacements of memory to reduce the computation time.

We discuss next this module on the basis of a run depicted in Figure 2, where the system drove the vehicle until the destination in a partially dynamic environment. We show in Figure 2a the complete trajectory of the experiment and the scans integrated using the odometry. This odometry information is so bad that successive scans cannot be used for the avoidance of non visible obstacles at each moment, and after some meters the robot is completely lost. Figure 2b shows the grid (400×400 cells and $0.05m$ each cell) computed by the model builder using the previous information. Notice how the grid reflects information of obstacles non detected with the last scan (since they are stored in the grid that maintains the robot in the center), which can be used for obstacle avoidance (Figure 2c), and how the model is suitable for planning purposes. There are also other issues to stand out: (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 (although it is not evident from the Figures, in the free space of the room there was a person moving) and (iii) the spurious measures are eliminated from the grid as new measures 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.

With respect to the functional and computational aspects of this module, with a grid of these characteristics we construct a model that represents an environment of 20×20 meters around the robot (portion of the environment sufficiently large to include the goal where we have to drive the robot). The module takes about 0.02sec (enough to work in real time).

B. Planning Module

This module uses a motion planner to obtain tactical information to avoid the trap situations and the cyclic motions. The idea behind our planner is to compute a portion of the space (that we call tunnel) that contains at least one path to the goal, but not to compute an analytical path as many classical planners do. This is because the course of the tunnel contains enough tactical information to avoid the trap situations, and an explicit path to the goal is not required. We illustrate the idea in Figure 3b, where the course of the tunnel that joins the initial and final configurations is enough to avoid the U-shape obstacle in between the initial and goal locations.

The planner constructs progressively a graph of connectivity, where the nodes are points in the free space and an arc between nodes means that at least exists one path between them. The process starts from the initial location

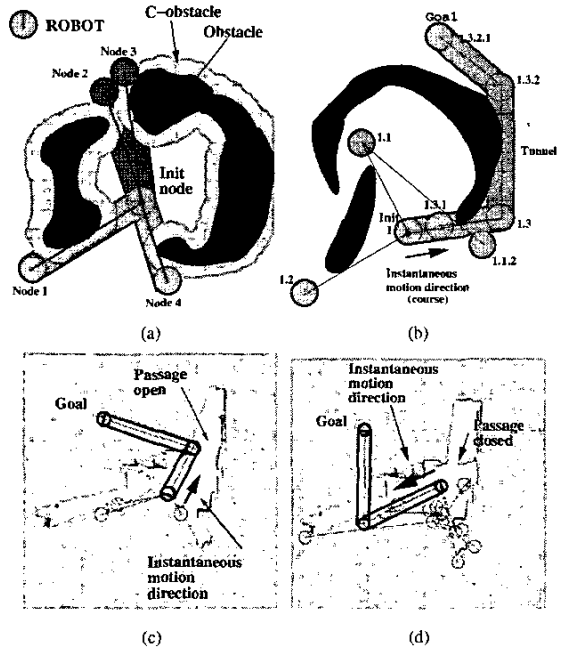


Fig. 3. These Figures show the performance of the planning module. (a) The expansion of a node, (b) computation of the tunnel of free space and course to follow in a scenario with a U-shape obstacle, (c) and (d) similar but in a real experiment.

and expands a node as follows: (i) from the position of the node we calculate the directions where there is a discontinuity between obstacles or where an obstacle ends, and we place a node there (as long as that space was not occupied by another node). Then, we compute for each new node whether it is accessible from the current node. More precisely, we calculate the existence of a tunnel of free space that contains a path that joins both positions (procedure detailed in [19]). If it exists, we incorporate this new node to the graph. For example in Figure 3a the tunnel in the configuration space for the nodes 1 and 4 is not blocked and thus they are accessible from the initial node (there is at least one path that joins both configurations). On the other hand, the tunnels corresponding to the nodes 2 and 3 are blocked, because the robot does not fit between these obstacles. (ii) We select the following node to expand by computing the distance covered from the initial location to this node plus the distance to the goal (cost). The process ends once the goal location is reached.

Figure 3b shows the incremental construction of the graph and the tunnel that joins the initial and goal locations. From the tunnel we obtain two types of information: first, if it is possible to reach the goal from the present position (since if the path exists the algorithm finds the tunnel that contains it). Secondly the *tactical motion direction* (main course of the first part of the tunnel). 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.

Figures 3c,d depict an experiment to show how the planner works over the available model and avoids one trap situation. Initially, in order to reach the goal the planner computed a course that aims towards a passage (Figure 3c). Suddenly, the passage was closed creating an end-zone (Figure 3d). However, the planner computed the new tunnel that pointed the way to the exit (backwards). Following this course the reactive method easily drove the vehicle out if this situation. The computation time of this algorithm is very dependent on the structure of the scenario since it guides the expansion of the nodes. In our typical indoor scenarios it works at medium rates of 0.03sec (enough for real-time).

C. Reactive Module

The ND+ method [21] is an improvement of the ND method [19], which is based on a methodology to design behaviors denominated the situated - activity paradigm (see [3] for a collection of works in this direction). In order to use this methodology, initially we describe a set of situations to represent the navigational problem and how to act in each of them (actions). Here, the situations represent an abstraction of all the cases between positions of the robot, obstacles and goal position (navigational situations). In addition, for each of these cases we associate a motion law (action). During the execution phase, at every moment we use information available of the obstacles and the position of the robot and the destination to identify one of these situations. Next the corresponding action is applied to compute the motion. This motion is executed by the robot and the process restarts.

The definition of the situations is based on criteria such as the security and entities that depend on the structure of the environment (e.g. areas of motion). The situations are represented using a binary decision tree, so that only one of them is chosen at every moment, and the motion laws are designed to obtain the behavior desired in each navigational situation.

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 [19] for a discussion on this topic). As it will be illustrated in the experimental results, these properties are 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 works at more than $1000Hz$, thus the reaction to the

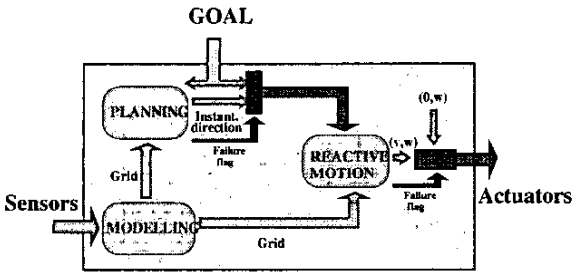


Fig. 4. This Figure depicts the integration architecture.

evolution of the scenario is very rapid and it can be used when required without imposing a significant time penalty.

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. The architecture has a synchronous planner - reactor configuration, where both parts use the model constructed in execution time (Figure 4). The functionality of the modules is the model construction, the computation of the tactical motion direction (to guide the reactive method), and the motion command generation.

There are situations where the modules produce failures which are managed by the architecture:

- Exception in the planning module: in some circumstances the planner does not find a solution, either because it does not exist (for example when the goal falls upon an obstacle) or because the module takes to much time and a time out is launched.
- Exception in the reactive module: the robot is completely surrounded by obstacles when there are not areas of motion (internal piece of information of the ND+ method), and thus the robot cannot progress.

In both cases, we set flags to carry out strategies that allow to close the control loop (Figure 4). In the first case, the reactive module directly uses the goal location instead of the information of the planner, and in the second one the robot stops and turns on itself (this behavior updates the model in all the directions hoping that a new passage is open).

The modules are executed in the following sequence: model - planner - reactor, dictated by the flow of data between modules. This flow is unidirectional, from the model module towards both the planner and the reactive module (with bandwidth of $\simeq 200\frac{kbytes}{sec}$) and from the planner towards the reactive module (with bandwidth of $\simeq 10\frac{bytes}{sec}$). The modules assure their time constraints to work synchronously with the sensor rate 0.25sec. We have assigned time-outs of (0.05, 0.08 and 0.02sec) to each module so that we always close the motion control loop (the maximum execution time is 0.15sec).

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 with a motion command available (there are no dead states). Furthermore, the modular structure of the system allows to replace the different modules easily, since the functional and computational aspects and their interfaces are clearly specified.

IV. Experimental Results

For experimentation, we used a commercial wheelchair that we have equipped with a SICK laser and with two onboard 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 $(1.2\times0.7meters)$ 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).

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, the vehicle transports humans that do not accept the abrupt movements and 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 not known, 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 covers 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 5a). During the motion, a person closed the right leaf of the door so that the wheelchair did not fit. Quickly the vehicle modified its way returning backwards (snapshot 2 of Figure 5a) in order to find the exit. During this passage the robot avoided collisions with the furniture and a person who moved bothering the normal progression of the vehicle (snapshot 3 of Figure 5a). In the centre of the office, the chair detected a very narrow door but sufficiently wide

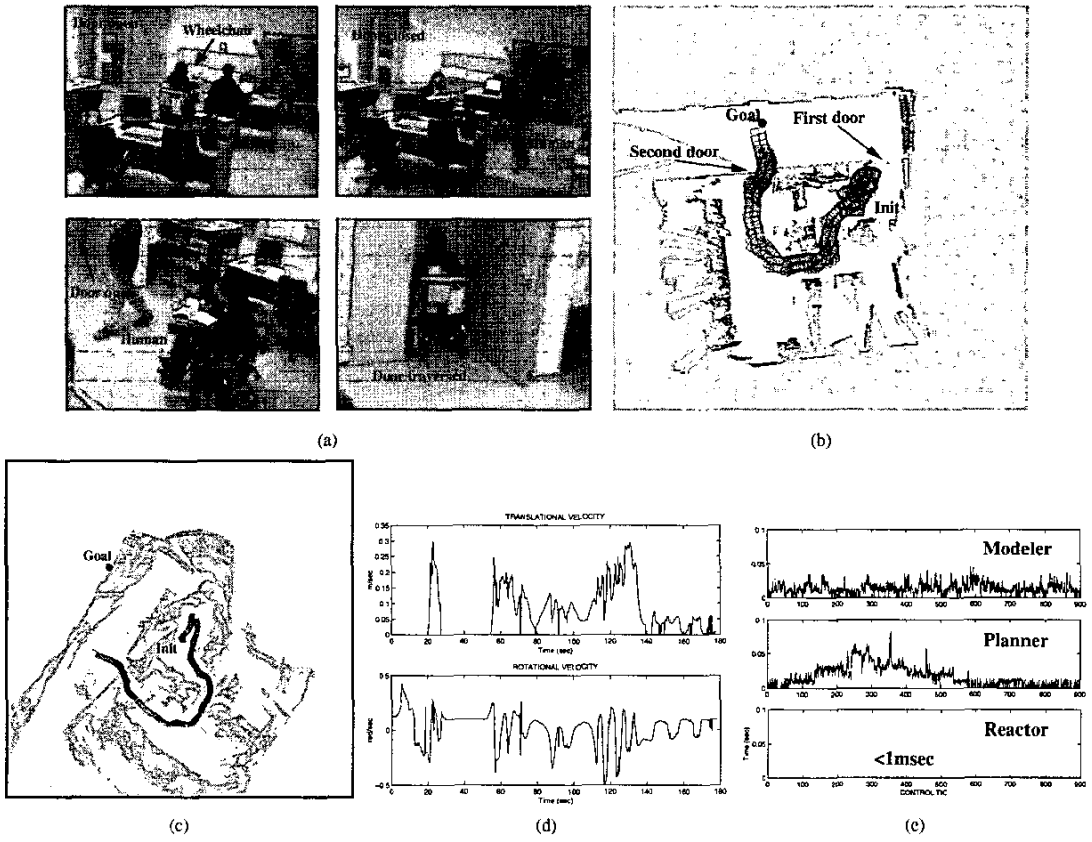


Fig. 5. These Figures show one experiment where the wheelchair drove a human out of the office. (a) Some snapshots of the experiment, (b) the model built during the experiment and the vehicle trajectory, (c) real laser data and trajectory using the odometry, (d) motion commands and (e) computation time of each module.

to fit in, thus the vehicle moved towards this door and maneuvered until crossing it and leaving the office reaching the final position (snapshot 4 of Figure 5a).

The data obtained from the laser and the odometry during the experiment are shown in Figure 5c, from which the main conclusion is that the odometry is quite bad and hardly could be used to deliberate or to compute motion. Nevertheless, the modelling module manages this information and constructs a reasonable model (Figure 5b). This is because the scan matching technique improves the odometry so that the information is properly integrated in the grid, and the vehicle approaches nearer to the destination. In addition, 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 benefits of this model are shared by the planning and the reactive modules, because there is a model available to compute courses to follow and the information of the non visible obstacles from the present position is also available (this case is understood when the door was crossed, snapshot 4 of Figure 5a), 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 5a). The course of the planner, previously pointing towards the door, rapidly changed aiming backwards (that directed the motion towards the outside of the end zone). This part of the run is depicted step by step in Figure 3c,d. This system avoids the trap situations and the cyclic behaviors by computing the course in each iteration.

The reactive module computed the collision free motion during the complete experiment taking into account the geometric, kinematic and dynamic constraints of the vehicle [18]. 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 5a)]. In addition, during the passage, 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 5d and the vehicle path in Figure 5b), 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.

We show the computation time of each module in the Figure 5e. The average execution time is 0.02sec for the model module, 0.03sec for the planner and lower than 0.001sec for the reactive method. With this rates all the modules worked synchronously within the cycle of the sensor, and none of the time outs were launched.

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 which is made up of three modules: a model constructor, a planning method and a reactive navigation method. Although some of these techniques derive from already existing works, the main contribution here is their integration to compose a complete system. The synergy of these modules carries out the navigation task achieving a higher level of performance and robustness.

The advantage of the system is that it is able to move vehicles in realistic scenarios, which are usually complicated, there is very little room to maneuver, are highly dynamic or create the well-known trap situations. We have demonstrated the usage of the system with a wheelchair vehicle under these work conditions.

VI. Acknowledgement

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

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

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

[15] B. H. Krogh and C. E. Thorpe. Integrated Path Planning and Dynamic Steering control for Autonomous Vehicles. In *IEEE Int. Conf. on Robotics and Automation*, pages 1664–1669, San Francisco, USA, 1986.

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

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

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

[19] 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.

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

[21] J. Minguez, J. Osuna, and L. Montano. A divide and conquer strategy to achieve reactive collision avoidance in troublesome scenarios. In *International Conference on Robotics and Automation*, Minessota, USA, 2004.

[22] 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.

[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] 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.