A Robotic Wheelchair Roaming in a Railway Station

E. Prassler, J. Scholz
Research Institute for
Applied Knowledge Processing (FAW)
P.O. Box 2060, D-89010 Ulm
Germany
{prassler,scholz}@faw.uni-ulm.de

P. Fiorini
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
USA
fiorini@jpl.nasa.gov

Abstract

In this paper we describe the hardware design and the control, and navigation system of our robotic wheelchair MAid (Mobility Aid for Elderly and Disabled People). Following the advice of disabled people and physicians we did not set out to re-invent and re-develop the set of standard skills of so-called intelligent wheel-chairs, such as FollowWall, FollowCorridor, PassDoorway, *which are commonly described in the literature. In spite of their disability handicapped people are often well capable of navigating their wheelchair along a corridor and actually eager to do it. Instead, we focused on maneuvers which are very burdensome because they take a long time and require extreme attention. One of these functions is deliberative locomotion in rapidly changing, large-scale environments, such as shopping malls or concourses of airports or railway stations, where tens or hundreds of people and objects are moving around. MAid's performance was tested in the central station of Ulm during rush-hour, and in the exhibition halls of the* Hannover Messe '98, *the biggest industrial fair worldwide. Meanwhile, MAid has survived more than 36 hours of testing in those environments. To our knowledge this is the first system among robotic wheel-chairs and mobile robots to have achieved a similar performance.*

1 Introduction

In this paper we describe a robotic wheelchair MAid (Mobility Aid for Elderly and Disabled People). MAid's task is to support and transport people with severely impaired motion skills such as, for example, paraplegia, multiple sclerosis, poliomyelitis, or muscular dystrophy, and to provide them with a certain amount of autonomy and independence. The system is based on a commercial wheelchair, which has been equipped with an intelligent control and navigation system.

Robotic wheel-chairs have been developed in a number of research labs (see our review in Section 2). The common set of functions provided by most of those systems consists of *AvoidObstacle FollowWall*, and *PassDoorway*. In conversations with disabled and elderly people, and with physicians we learned that not all of these functions are of equal interest for people with motion impairment. Particularly *FollowWall* and *PassDoorway* are maneuvers which most disabled people still want to execute themselves provided they have the necessary fine motor control.

Following this advice, in our work we focused on different types of motion skills. Our system has two modes of operation, a semi-autonomous and a fully autonomous mode. In the semi-autonomous mode the user can command MAid to execute local maneuvers in narrow, cluttered space. For example, the user can command MAid to maneuver into the cabin of a restroom for handicapped people. Maneuvers in narrow, cluttered space require extreme attention and often lead to collisions, particularly if the patient lacks sufficient fine motor control. We denoted this type of maneuver in small, narrow areas as NAN (narrow area navigation), and the implementation of this capability is described in [9, 11].

In the second mode, MAid navigates fully autonomously through wide, rapidly changing, crowded areas, such as concourses, shopping malls, or convention centers. The algorithms and the control system which enable MAid to do so are described in the following. We denoted this latter type of motion skill as WAN (wide area navigation). The only action the user has to take is to enter a goal position. Planning and executing a trajectory to the goal is completely taken care by MAid.

MAid's performance was tested in the central station of Ulm during rush-hour and in the exhibition halls of the *Hannover Messe '98*, the biggest industrial fair worldwide. Altogether, MAid has so far survived more than 36 hours of testing in public, crowded environments with heavy passenger traffic. To our knowledge there is no other robotic wheelchair and no other mobile robot system which can claim a comparable performance.

Note that at first sight the two types of motion skills, NAN and WAN, have little in common with the navigation skills of other intelligent wheel-chairs. Quite the opposite is the case. In terms of its performance WAN can be seen as a superset of functions such as *AvoideObstacle* or *FollowWall.* When WAN is activated and a destination at the opposite end of a hallway is specified then MAid will automatically show a wall following behavior and at the same time avoid obstacles, although there is no explicit implementation of such a behavior in the WAN module. Likewise, passing a door or docking at a table are typical instances of NAN maneuvers.

2 Related Work

In recent years there has been the development of several intelligent wheel-chairs. A first design concept for a self-navigating wheelchair for disabled people was proposed by Madarasz in [4]. The system NavChair is described in [1]. NavChair's most important function is automatic obstacle avoidance. Other functions include wall following and passing doorways. Hoyer and Hölper [8] present a modular control architecture for an omni-directional wheelchair. A low-level control unit is in charge of the operation of the sensor apparatus, the actual motion of the vehicle and the operation of a manipulator. A high-level PC/UNIX based planning module consists of a path and a task planner to execute task oriented commands. A hybrid vehicle RHOMBUS for bedridden persons is described in [5]. RHOMBUS is a powered wheelchair with an omni-directional drive which can be automatically reconfigured such that it becomes part of a flat stationary bed. Mazo et al. [6] describe an electrical wheelchair which can be guided by voice commands. The wheelchair recognizes commands such as *Stop, Forward, Back, Left, Right, Plus, Minus.* Miller and Slack [7] designed the systems Tin Man I and Tin Man II. Both systems were built on top of a commercial pediatric wheelchair. Tin Man I has three operation modes: *human guided with obstacle override, move forward along a heading,* and *move to* (x,y). These functions were substantially extended in Tin Man II. Tin Man II capabilities include *Backup, Backtracking, Wall Following, Passing Doorways, Docking* and others. Wellman [12] proposes a hybrid wheelchair which is equipped with two legs in addition to the four regular wheels, which enable the wheelchair to climb over steps and move through rough terrain.

3 Hardware Design

Our system MAid (see Fig. 1) is based on a commercial electrical wheelchair type SPRINT manufactured by MEYRA GmbH in Germany. The wheelchair has two differentially driven rear wheels and two passive castor front wheels. It is powered by two 12 V batteries (60 Ah) and reaches a maximum speed of 6 km/h. The standard vehicle can be manually steered by a joystick.

The hardware core of MAid's navigation system is an industrial PC (Pentium 166MHz) which serves as on-board computer. The computer is controlled by the real-time operating system QNX. MAid is furthermore equipped with a variety of sensors for environment perception, such as collision avoidance, and position estimation. MAid's sensor apparatus includes

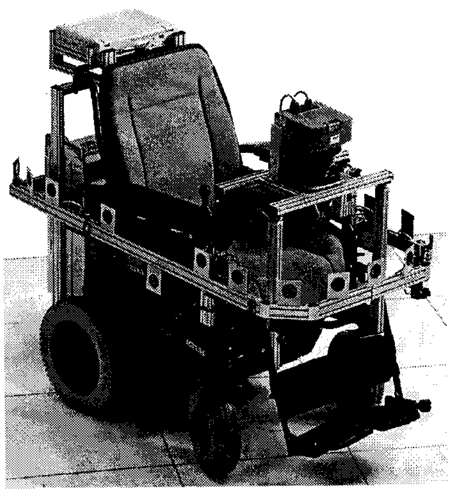
Figure 1: The robotic wheelchair MAid

- a *dead-reckoning system* consisting of a set of wheel encoders and a optical fiber gyroscope (Andrew RD2030),
- a *modular sonar system* consisting of 3 segments each equipped with 8 ultra-sound transducers and a micro-controller, mounted on an aluminum frame which can be opened to enable the user to sit in the wheelchair,
- two *infrared scanners* (Sharp GP2D02 mounted on servos) for short range sensing, and
- a SICK 2D *laser range-finder* PLS 200 mounted on a removable rack.

The dead-reckoning system, which integrates over the distance traveled by the vehicle and over its orientation changes, provides elementary position and orientation information. This information is rather inaccurate with errors accumulating rapidly over the traveled distance but it is available at low cost and at all times.

The sonar system and the laser range finder are the sensors which MAid uses to actually perceive the surrounding environment. This perception has the form of two dimensional range profiles and gives a rather coarse picture of the environment. From the range profiles MAid extracts the basic spatial structure of the environment. In particular, MAid uses these range data are detect and to track moving objects in its surroundings.

Except for the laser range finder, MAid's sensors are connected to, and communicate with, the on-board computer using a *field bus.* The interface between these devices and the field bus is implemented by a number of micro-controllers (68HC11). Due to the high data rates of the laser range-finder, this device is directly connected to the on-board computer by a serial port. The motion commands computed by the navigation system are also sent over the field bus to the motion controller, which powers the wheel motors.

4 Control Architecture

MAid has a hierarchical control architecture consisting of three levels: a basic control level, a tactical level, and a strategic level. The parts of this control system which contribute to MAid's capability of navigating in a wide, crowded, rapidly changing area (WAN) are shown in Fig. 2.

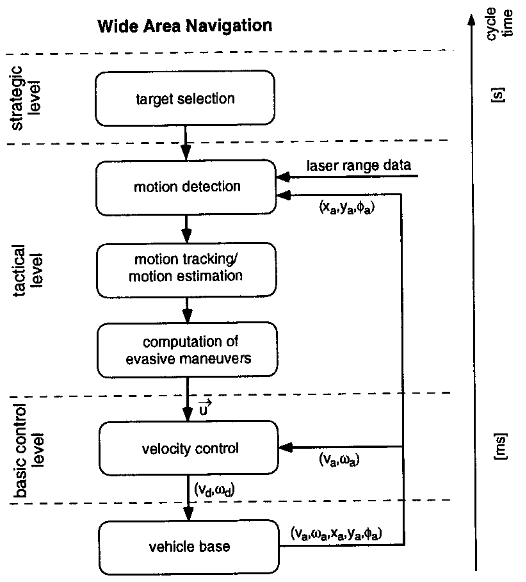

Figure 2: Software architecture of MAid's WAN module

On the basic control level we compute the values of the control variables which put the vehicle into motion. The *velocity control* module on this basic control level receives as input a velocity vector $\vec{u}$ describing the target velocity and heading and the actual values of the translational and rotational (v_a, ω_a) of the vehicle. The velocity vector is converted into target values for the translational and rotational velocity. Out of these target values and the actual values provided by the vehicle's dead-reckoning system the velocity controller computes appropriate correction values which are then fed to the motor controllers.

On the tactical level, which essentially forms the core of the WAN module we have three submodules, a *motion detection* module, a module for *motion tracking and estimating the velocities and headings* of the objects in MAid's vicinity, and a module for *computing evasive maneuvers.*

In the continuous stream of range data provided by its range-finder MAid tries to detect the objects in its environment and to identify which of these objects are stationary and which are in motion (see [10] for more details). From the stream of range data MAid further derives estimates for the motion direction and velocity of the objects by extrapolating their past motion trajectory and velocity.

Based on these predictions and on its own motion direction and velocity MAid then determines if it is moving on a collision course with one or several of the moving objects. After an analysis of so-called *Velocity Obstacles* [3] MAid computes an avoidance maneuver, which is as close to its original heading as possible but does not lead to a collision with the objects moving in the vicinity of MAid.

MAid's main task while it navigates in a wide, crowded, rapidly changing area is to reach a specific goal at some distance from its present position. In the current design it does not pursue any more complex, further reaching plans such as visiting a sequence of intermediate goals. Accordingly, the strategic level consists of the selection of the next goal, which is left to the user. At a later point, the strategic level will be expanded by a path planner, for example, which will provide the WAN module with a sequence of intermediate goals.

5 Navigation in Crowded Environments

In this section, we describe the methods and components which constitute MAid's tactical level and essentially enable it to navigate in a wide, crowded, rapidly changing area (WAN).

5.1 Motion Detection and Tracking

A rather obvious approach to identify changes in the surrounding environment is to consider a sequence of single observations and to investigate where these observations differ from each other. A discrepancy between two subsequent observations is a strong indication of a potential change in the environment. Either an unknown object has been discovered due to the self-motion of the observer or an already discovered object has moved by some distance. In the following sections we discuss how this simple idea can be used in a fast motion detection and tracking algorithm.

Representations of Time-varying Environments

A very efficient and straightforward scheme for mapping range data is the *occupancy grid representation* [2]. This representation involves a projection of the range data on a two-dimensional rectangular grid, where each grid element describes a small region in the real-world.

While investigating the performance of existing grid based mapping procedures, we noticed that most of the time was spent for mapping free space. Particularly, the further away the observed objects were, the more time it costs to map the free space between the sensor and the object. Also, before a range image could be assimilated into a grid, the grid had to be completely initialized, that is, each cell had to be set to some default value.

For grids with a typical size of several tens of thousands of cells these operations became quite expensive.

To avoid these time consuming operations, we devised an alternative representation in which we map only the cells observed as occupied at time t, whereas all other cells in this grid remain untouched. We call this representation a *time stamp map.*

Compared to the assimilation of a range image into an occupancy grid the generation of a time stamp map is rather simplified. Mapping a range measurement involves only one single step, i.e.the cell coinciding with the range measurement is assigned a time stamp t. This stamp means that the cell was *occupied at time t.* No other cell is involved in this operation. Particularly, we do not mark as *free* any cell which lies between the origin of the map and the cell corresponding to the range measurement.

The time variation of the environment is captured by the sequence TSM_t, $TSM_{t-1}, \dots,$ TSM_{t-n} of those time stamp maps. An example of such a sequence is shown in Fig. 5.1 a) - c). The aligned maps are shown in Fig. 5.1 d). The assimilation of a range image into a 200×200 time stamp map takes 1.5 ms on a Pentium 166Mhz.

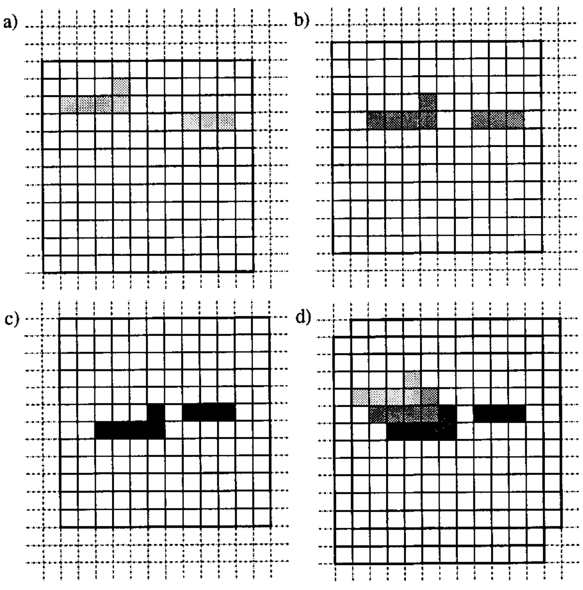

Figure 3: A sequence of time stamp maps describing a toy-environment.The age of the observation is indicated through different gray levels. The more recent the observation the darker is the gray level which marks the object contours. Figure d) shows the aligned maps.

An Approach to Fast Motion Detection

Motion detection in a sequence of time stamp maps is based on a simple heuristic. We consider the set of cells in TSM_t which carry a time stamp t (*occupied at time t*) and test whether the corresponding cells in TSM_{t-1} were occupied too, i.e., carry a time stamp $t-1$. If corresponding cells in TSM_t, TSM_{t-1} carry time stamps t and $t-1$, respectively, then we interpret this as an indication that the region in the real world, which is described by these cells has been occupied by a stationary object. If, however, the cells in TSM_{t-1} carry a time stamp different from $t-1$ or no time stamp at all, then the occupation of the cells in TSM_t must be due to a moving object.

Motion Tracking and Estimation of Object Trajectories

Since there is no analytical model of human purposive locomotion, which would allow us to make inferences about the motion of a person over longer distances, the best we can do to track a moving person in environment such as a crowded concourse in a railway station is to collect information about its past motion and to extrapolate this past motion into the near future, if necessary. For this purpose we consider the sequence of recent sensor images and extract the information about motion direction, velocity, or acceleration describing the motion history of the moving objects from the spatial changes which we find in the mappings of these sensor images.

We assume that the cells describing distinct objects are grouped into ensembles, and we also assume that these ensembles and their corresponding objects are classified either as *moving* or as *stationary* by the motion detection algorithm described above.

The first step in establishing the motion history of an object is to identify the object in a sequence of mappings. Once we have found this correspondence it is easy to derive the heading and the velocity of a moving object from its previous positions. To find a correspondence between the objects in the mappings of subsequent sensor images we use a nearest-neighbor criterion. This criterion is defined over the Euclidean distance between the centers of gravity of cell ensembles representing distinct objects. For each cell ensemble representing an object at time t we determine the nearest ensemble in terms of the Euclidean distance in the map describing the environment at the preceding time steps $t-1$.

If the distance to the nearest neighbor is smaller than a certain threshold then we assume that both cell ensembles describe the same object. The threshold depends on whether the considered objects and cell ensembles are stationary or moving. For establishing a correspondence between the two cell ensembles describing a stationary object we choose a rather small threshold since we expect the cell ensembles to have very similar shapes and to occupy the same space. Currently, we use a threshold of 30 cm for stationary objects. For a correspondence between the cell ensembles describing a moving object this value is accordingly larger. Here we use a threshold of 1 m which is approximately the distance which a person moving at fast walking speed covers between two sensor

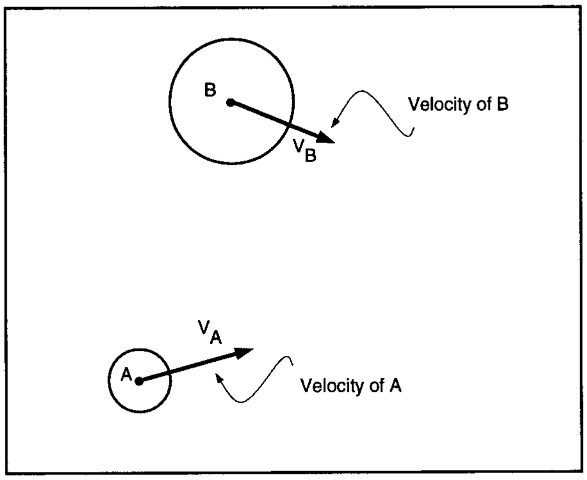

Figure 4: The mobile robot A and the moving obstacle B.

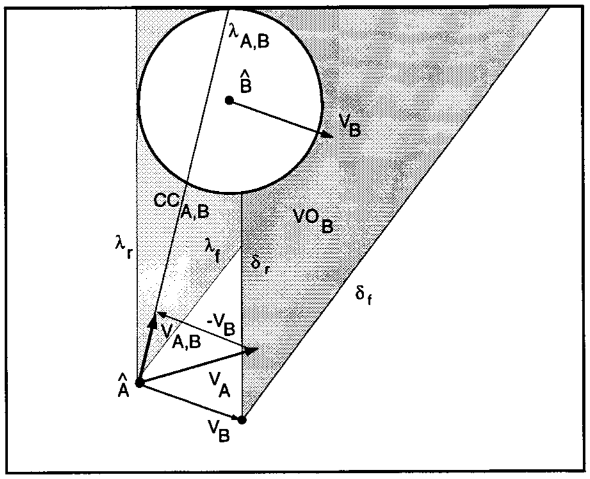

Figure 5: The Relative Velocity $\mathbf{v}_{A,B}$, the Collision Cone $CC_{A,B}$, and the Velocity Obstacle VO_B.

images. Note that on a Pentium 166MHz a complete cycle involving both detecting and tracking any moving objects takes approximately 6 ms.

5.2 Motion Planning Using Velocity Obstacles

In this section, we briefly summarize the concept of *Velocity Obstacle* (VO) for a single and multiple obstacles. For simplicity, we model the robotic wheelchair and the obstacles as circles, thus considering a planar problem with no rotations. This is not a severe limitation since general polygons can be represented by a number of circles. Obstacles move along arbitrary trajectories, and their instantaneous state (position and velocity) is estimated by MAid's sensors, as discussed earlier.

To introduce the Velocity Obstacle (VO) concept, we consider the two circular objects, A and B, shown in Figure 4 at time t_0, with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$. Let circle A represent the mobile robot, and circle B represent an obstacle. To compute the VO, we first map B into the *Configuration Space* of A, by reducing A to the point $\hat{A}$ and enlarging B by the radius of A to $\hat{B}$, and represent the state of the moving object by its position and a velocity vector attached to its center. Then, the set of colliding *relative* velocities between $\hat{A}$ and $\hat{B}$, called the *Collision Cone*, $CC_{A,B}$, is defined as $CC_{A,B} = \{\mathbf{v}_{A,B} \mid \lambda_{A,B} \cap \hat{B} \neq \emptyset\}$, where $\mathbf{v}_{A,B}$ is the relative velocity of $\hat{A}$ with respect to $\hat{B}$, $\mathbf{v}_{A,B} = \mathbf{v}_A - \mathbf{v}_B$, and $\lambda_{A,B}$ is the line of $\mathbf{v}_{A,B}$. This cone is the light grey sector with apex in $\hat{A}$, bounded by the two tangents λ_f and λ_r from $\hat{A}$ to $\hat{B}$, shown in Figure 5. Any relative velocity that lies between the two tangents to $\hat{B}$, λ_f and λ_r, will cause a collision between A and B. Clearly, any relative velocity outside $CC_{A,B}$ is guaranteed to be collision-free, provided that the obstacle $\hat{B}$ maintains its current shape and speed.

The collision cone is specific to a particular pair of robot/obstacle. To consider multiple obstacles, it is useful to establish an equivalent condition on the ***absolute*** velocities of A. This is done simply by adding the velocity of B, $\mathbf{v}_B$, to each velocity in $CC_{A,B}$ and forming the *Velocity Obstacle VO*, $VO = CC_{A,B} \oplus \mathbf{v}_B$ where $\oplus$ is the Minkowski vector sum operator, as shown in Figure 5 by the dark grey sector. The VO partitions the absolute velocities of A into *avoiding* and *colliding* velocities. Selecting $\mathbf{v}_A$ outside of VO would avoid collision with B. Velocities on the boundaries of VO would result in A grazing B.

To avoid multiple obstacles, we consider the union of the individual velocity obstacles, $VO = \cup_{i=1}^{m} VO_{B_i}$, where m is the number of obstacles. The avoidance velocities, then, consist of those velocities $\mathbf{v}_A$, that are outside all the VO's.

In the case of many obstacles, obstacles avoidance is prioritized, so that those with imminent collision will take precedence over those with long time to collision. Furthermore, since the VO is based on a linear approximation of the obstacle's trajectory, using it to predict remote collisions may be inaccurate, if the obstacle does not move along a straight line. By introducing a suitable *Time Horizon* T_h, we limit the collision avoidance to those occurring at some time $t < T_h$.

The Avoidance Maneuver

An *avoidance maneuver*, consists of a one-step change in velocity to avoid a future collision within a given time horizon. The new velocity must be achievable by the moving robot, thus the set of avoidance velocities is restricted to the velocities achievable by robot A at a given state over a given interval. This set of *reachable velocities* can be superimposed on the velocity obstacle VO, thus graphically representing the set of *reachable avoidance velocities*, RAV, available for an avoidance ma-

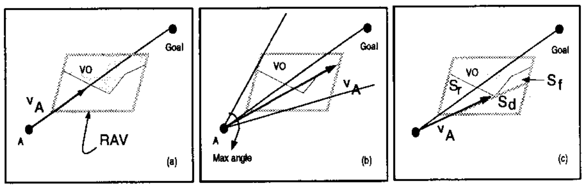

Figure 6: **a**: TG strategy. **b**: MV strategy. **c**: ST strategy.

neuver. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

It is then possible to choose the type of an avoidance maneuver, by selecting on which side of the obstacle the mobile robot will pass. As discussed earlier, the boundary of the velocity obstacle VO, $\{\delta_f, \delta_r\}$, represents all absolute velocities generating trajectories tangent to $\widehat{B}$, since their corresponding relative velocities lay on λ_f and λ_r. The possibility of subdividing the avoidance velocities RAV into subsets, each corresponding to a specific avoidance maneuver of an obstacle, is used by the robotic wheelchair to avoid obstacles in different ways, depending on the perceived danger of the obstacle.

Computing the avoidance trajectories

A complete trajectory for the mobile robot consists of a sequence of single avoidance maneuvers that avoid static and moving obstacles, move towards the goal, and satisfy the robot's dynamic constraints. The trajectory is generated in real-time, incrementally by selecting a single avoidance velocity at each discrete time interval, using some heuristics to choose among all possible velocities in the reachable avoidance velocity set RAV.

The heuristics can be designed to satisfy a prioritized series of goals, such as survival of the robot as the first goal, and reaching the desired target, minimizing some performance index, and selecting a desired trajectory structure, as the secondary goals. Choosing velocities in RAV (if they exist) automatically guarantees survival. Among those velocities, selecting the ones along the straight line to the goal would ensure reaching the goal, the TG strategy shown in Figure 6. Selecting the highest feasible velocity in the general direction of the goal may reduce motion time, the MV heuristics shown in Figure 6. Selecting the velocity from the appropriate subset of RAV can ensure a desired trajectory structure (front or rear maneuvers), the ST heuristics shown in Figure 6. It is important to note that there is no guarantee that any objective is achievable at any time. The purpose of the heuristic search is to find a "good" local solution if one exist.

In the experiments described in the following section, we used a combination of the TG and the ST heuristics, to ensure that the robotic wheelchair moves towards the goal specified by the user. When the RAV sets include velocity vectors aiming directly to the goal, the largest among them is chosen for next control cycle. Otherwise, the algorithm computes the centers of the RAV sets, and chooses the velocity corresponding to the center closest to the direction to the goal. This heuristics adds also an additional safety margin to the mobile robot trajectory, since the velocity chosen is removed from the boundary of its RAV set, thus accounting for unmodelled uncertainties on the obstacle shapes and trajectories.

6 MAid Roaming in a Railway Station

After MAid had successfully passed a number of laboratory experiments it was confronted with a real world environment, which was the concourse of the central station in Ulm, a hall of approximately $15 \times 40\,\mathrm{m}^2$. First test runs were conducted during the morning rush hours. We thought that this would represent the worst scenario MAid would ever have to face. In fact, after the arrival of a commuter train typically up to several hundred people moved through the concourse within two or three minutes. We counted up to 150 people crossing the concourse within about a minute. After two or three minutes however, the concourse was practically empty again, thus leaving not enough time for conducting experiments. The railway station manager, who was observing our experiments with great interest, finally told us that the ideal time for our tests would have been Friday noon, which does not exhibit the densest but the most continuous passenger traffic in the concourse. During the period between 11:00 am and 1:30 pm in fact, typically several tens of people stay and move around in the concourse, thus making it very suitable for the navigation experiments.

To test MAid's navigation performance we let it cross the concourse in arbitrary directions. MAid is put in motion by pushing the joystick shortly into the direction of the target location and by entering a travel distance. The wheelchair then starts moving in the desired direction as long as there is no collision imminent. If a collision with an object or a person is impending, MAid, while continuing its motion, determines a proper avoidance maneuver and follows this new direction until it can turn back and head again to the goal location. Snapshots of MAid's test runs in the crowded concourse are shown in Fig. 7. The passenger traffic in these images is moderately dense, which actually facilitated recording the pictures. When the passenger traffic became too dense, MAid occasionally simply stopped, and did what a human operator would have also probably done in that situation: it waited until the group of people blocking its way had passed and then it continued its journey.

During our experiments in the concourse MAid collided several times with objects. Usually these objects were bags or a suitcases lying on the floor invisible to MAid's laser range-finder and its sonar sensors. To

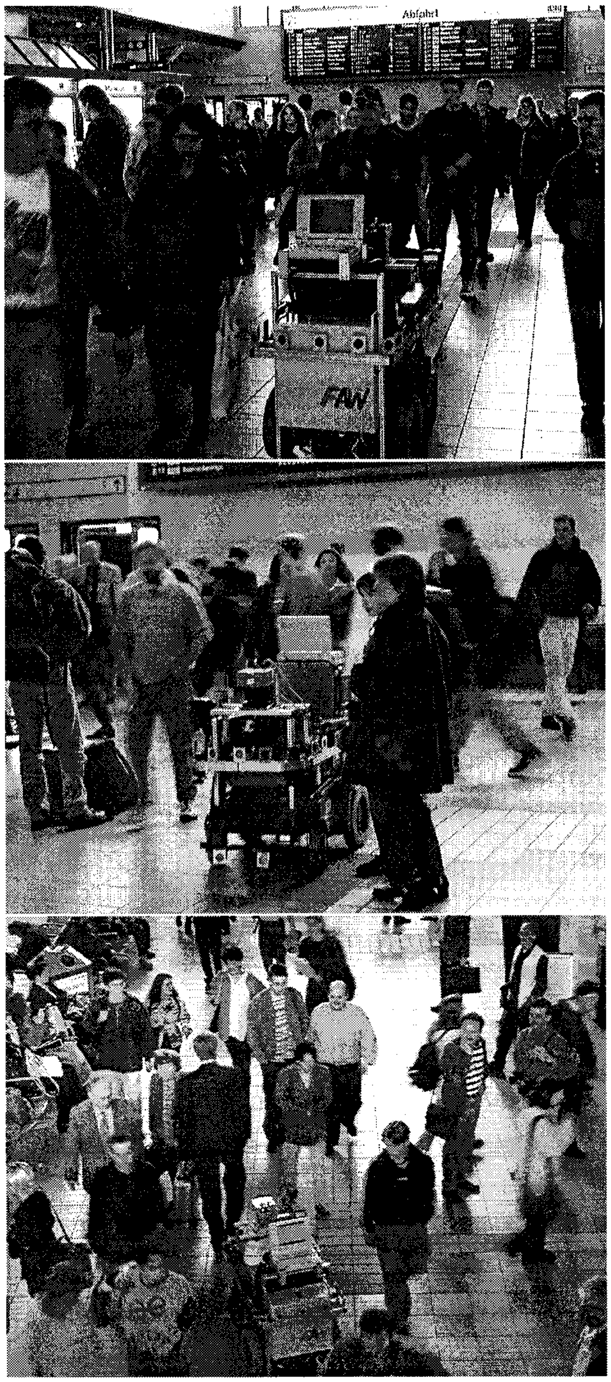
Figure 7: MAid traveling in the concourse of a railway station.

discover small obstacles in front of the wheelchair we mounted two extra sonar sensors to the foot rests of the wheelchair.

So far, MAid has survived about 18 hours of testing in the concourse of the central station in Ulm and we continue conducting experiments in this environment.

MAid was presented to a wider audience during the Hannover Fair '98. The Hannover Fair is the largest industrial fair worldwide. In Hannover, MAid drove through the exhibition halls for seven days between two and three hours per day at regular visiting hours. Altogether MAid has successfully navigated in crowded, rapidly changing environments for more than 36 hours.

Acknowledgment

This work was supported by the German ministry for education, science, research, and technology (BMB+F) under grant no. 01 IN 601 E 3 as part of the project INSERVUM. The development of the Velocity Obstacle approach was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] D.A. Bell, J. Borenstein, S.P. Levine, Y. Koren, and L. Jaros. An Assistive Navigation System for Wheelchairs Based upon Mobile Robot Obstacle Avoidance. In *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, 1994.

[2] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Electrical and Computer Engineering Department/Robotics Institute, Carnegie-Mellon University, 1989.

[3] P. Fiorini, Z. Shiller. Motion Planning in Dynamic Environments Using the Relative Velocity Paradigm. In *Proc. of the 1993 IEEE Int. Conf. on Robotics and Automation*, Atlanta, 1993.

[4] R.L. Madarasz, L.C. Heiny, R.F. Cromp, N.M. Mazur. The Design of an Autonomous Vehicle for the Disabled. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No.3, 1986.

[5] S. Mascaro, J. Spano, H. Asada. A Reconfigurable Holonomic Omnidirectional Mobile Bed with Unified Seating (RHOMBUS) for Bedridden Patients. In *Proc of the 1997 IEEE Int. Conf. on Robotics and Automation*, Albuquerque, 1997.

[6] M. Mazo, F.J. Rodriguez, J.L. Lazaro, J. Urena, J.C. Garcia, E. Santiso, P.A. Revenga, and J.J. Garcia. Wheelchair for Physically Disabled People with Voice, Ultrasonic and Infrared Sensor Control. *Autonomous Robots*, 2, 1995.

[7] D. Miller, M. Slack. Design and Testing of a Low-Cost Robotic Wheelchair Prototype. *Autonomous Robots*, 2, 1995.

[8] H. Hoyer, R. Hölper. Open Control Architecture for an Intelligent Omnidirectional Wheelchair. In *Proc. of the 1st TIDE Congress*, Brussels, IOS Press, 1993.

[9] E. Prassler, J. Scholz, M. Strobel. MAid: Mobility Assistance for Elderly and Disabled People. In *Proc. of the 24th Int. Conf. of the IEEE Industrial Electronics Soc. IECON'98*, Aachen, Germany, 1998, (to appear).

[10] E. Prassler, J. Scholz, E. Elfes. Tracking People in a Railway Station During Rush-Hour (submitted).

[11] M. Strobel, E. Prassler, D. Bank. Navigation of non-circular mobile robots in narrow, cluttered environments (in preparation).

[12] P. Wellman, V. Krovi, V. Kumar. An Adaptive Mobility System for the Disabled. In *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, 1994.