Navigating a Robotic Wheelchair in a Railway Station During Rush-Hour

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, the **control** and **navigation** system, and our preliminary experiments with the **robotic wheelchair MAid (Mobility Aid for Elderly and Disabled People)**. MAid's general task is to transport people with **severely impaired motion skills** such as, for example, paraplegia, multiple sclerosis, poliomyelitis, or muscular dystrophy. 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 wheelchairs, such as *FollowWall, FollowCorridor, PassDoorway* which are commonly described in the literature. These maneuvers do not always require fine motion control and disabled people, in spite of their disability, are often well capable of navigating their wheelchair along a corridor and actually eager to do it. In our work we focused instead 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, entry halls of theaters, and concourses of airports or railway stations, where tens or hundreds of people and objects are moving around. This function was not only acknowledged as being very useful but also very entertaining, because MAid often had to work very hard to find its way through a crowd of people. 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 survived more than 36 hours of testing in public, **crowded environments** with **heavy passenger traffic**. To our knowledge this is the first system among robotic wheelchairs and mobile robots to have achieved a similar performance.

1 Introduction

Nowadays, the freedom and capability to move around unrestrictedly and head for almost any arbitrary location seems to be an extremely valuable commodity. Mobility has become an essential component of our quality of life. A natural consequence of this appreciation of the good mobility is the negative rating of the loss of mobility caused, for example, by a disease or by age. The loss of mobility represents not only the loss of a physiological function, but often a considerable social descent. People with severely impaired motion skills have great difficulties to participate in a regular social life. Not seldom a loss of mobility leads to a loss of contacts to other non-disabled people or makes it at least difficult to establish such contacts. A loss of mobility, may it be due to an injury or to advanced age, is always accompanied by a loss of autonomy and self-determination, it creates dependence and in extreme cases it may even affect the individual intimacy and dignity.

The loss of one's mobility may be seen as a difficult individual fate. However, there are two aspects which may make it not only into an individual but also into a general problem of our society. First, the average age in western societies is increasing dramatically. As a natural consequence the number of people suffering from severe motion impairment will increase too. At the same time, we can observe an equally dramatic increase of the expenditures for health care and nursing and furthermore a reduction of nursing staff in order to limit the cost explosion. The results of these developments are foreseeable: the quality of health care will decay, individual care will become still more expensive and less affordable for people with medium and lower income, elderly will then be sent to nursing homes much earlier than nowadays in order to get sufficient care.

A way out of this unpleasant development may be through the development of robotic technologies. Many activities which a person with severe motion impairment is unable to execute may become feasible by using robot manipulators and vehicles as arms and legs, respectively. Lifting things, carrying and manipulating things and moving around in ones own home this way becomes feasible without the assistance and help of a nurse. People with motion impairment get back a certain amount of autonomy and independence and can stay in their familiar environment. Expenditures for nursing personnel or an accommodation in a nursing home can be avoided or at least limited.

In this paper we describe a robotic wheelchair MAid (Mobility Aid for Elderly and Disabled People) whose task is to transport people with severely impaired motion skills 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 wheelchairs have been developed in a number of research labs (see our review in Section II.) 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 [12, 15].

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 capability of navigating in rapidly changing environments was not only acknowledged as being very useful but also very entertaining. MAid often had to work very hard to find its way through a crowd of people and our test pilots were often very curious to see what MAid would do next, bump into a passenger - very rarely it did - or move 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. 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, which MAid is capable

of, have little in common with the navigation skills of other intelligent wheelchairs. 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.

The rest of this paper is organized as follows. In the next section we give an overview of the state of the art in the development of robotics wheelchairs. MAid's hardware design is described in Section 3. In Section 4, we then describe the software architecture and the algorithms which enable MAid to navigate in a wide, rapidly changing, crowded environment. Note that although MAid's capability to navigate in narrow, partially unknown, cluttered environment is mentioned several times below, the focus in this paper is on MAid's WAN skill. We will not go into the details of MAid's NAN skill, presented in [12, 15].

2 Related Work

In recent years there has been the development of several intelligent wheelchairs. A first design concept for a self-navigating wheelchair for disabled people was proposed by Madarasz in [8]. The vehicle described there used a portable PC (320 KB of memory) as on-board computer. The sensor equipment of the wheelchair included wheel encoders, a scanning ultrasonic range-finder and a digital camera. The system was supposed to navigate fully autonomously in an office building. To find a path to its destination it used a symbolic description of significant features of the environment, such as hallway intersections or locations of offices. The path computed by the path planner consisted of a sequence of primitive operations such as *MoveUntil* or *Rotate*.

In [3], the system NavChair is described. NavChair's on-board computer is also a portable IBM compatible PC. An array of 12 Polaroid ultrasonic sensors at the front of the wheelchair is used for obstacle detection and avoidance. NavChair's most important function is automatic obstacle avoidance. Other functions include wall following and passing doorways.

Hoyer and Hölper [7] present a modular control architecture for an omni-directional wheelchair. The drive of this system is based on Meccanum-wheels. The wheelchair is equipped with ultrasonic and infrared sensors and a manipulator. 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 the manipulator. This control unit is realized on an VME-Bus-system using pSOS+. 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 [9]. 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. The bedridden person does not have to change seating when transferring between the chair and bed.

Mazo et al. [10] describe an electrical wheelchair which can be guided by voice commands. The wheelchair recognizes commands such as *Stop, Forward, Back, Left, Right, Plus, Minus* and turns them into elementary motion commands. The system also has a control mode *Autonomous*. In this mode the wheelchair follows a wall at a certain distance.

Miller and Slack [11] designed the system Tin Man I and its successor Tin Man II. Both systems were built on top of a commercial pediatric wheelchair from Vector Wheelchair Corporation. Tin Man I used five types of sensors, drive motor encoders, eight contact sensors used as whiskers, four IR proximity sensors distributed along the front side of the wheelchair, six sonar range sensors, and a flux-gate compass to determine the vehicle's orientation. Tin Man I had 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.

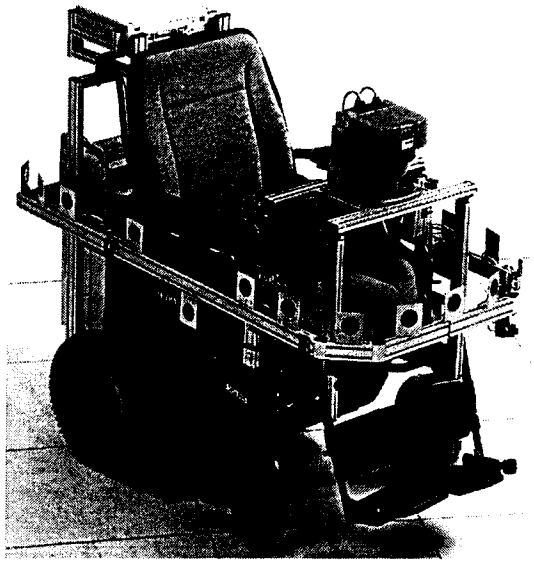

Figure 1: The robotic wheelchair MAid

Wellman [16] proposes a hybrid wheelchair which is equipped with two legs in addition to the four regular wheels. These legs should enable the wheelchair to climb over steps and move through rough terrain. A computer system consisting of a PC 486 and a i860 co-processor for the actuator coordination is used to control the wheelchair.

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 goal of the work presented here was to develop a complete navigation system for a commercial wheelchair, such as SPRINT, which would enable it to automatically maneuver in narrow, cluttered space as well as in crowded large-scale environments. The hardware core of the navigation system developed for the task 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 equipped with a variety of sensors for environment perception, such as collision avoidance, and position estimation. In particular, MAid's sensor apparatus includes the following devices:

- 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,
- 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

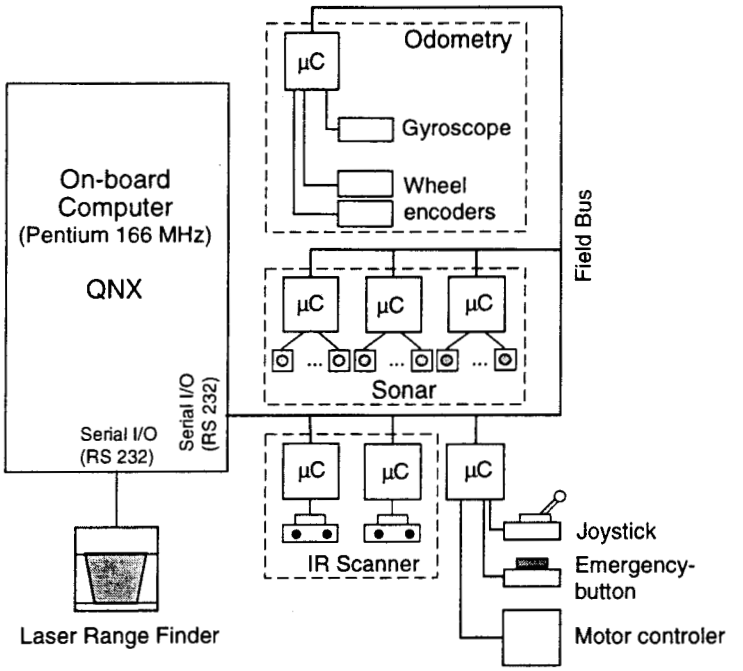

Figure 2: Hardware architecture of MAid's control system

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, which is then used for two purposes: the *avoidance of stationary and moving objects*, and the *estimation of MAid's position* in the environment.

For the latter, we apply an Extended Kalman filter, provided we have a description of the environment. Based on such a description, on a model of MAid's locomotion, and its last position this filter produces a set of expectations regarding MAid's sensor readings. The deviation between these expectations and the true sensor readings is then used to compute correction values for MAid's estimated position.

It should be mentioned that in a wide, crowded, rapidly changing, mostly unknown environment there is little advantage in using a Kalman filter since one of its essential ingredients, namely the *a priori* description of the environment is not available. For the navigation in such an environment, we have to rely exclusively on the position information provided by the dead-reckoning system.

While all other components of MAid's navigation system are low cost or can at least be substituted by cheaper components without reducing MAid's performance, the laser range-finder is undoubtedly an expensive sensor (approx. US$ 4000). However, as we have argued in [13, 14] there is no other sensor which is equally suited for detecting and tracking a large number of moving objects in real-time. This function in turn is essential for the navigating in wide, crowded, rapidly changing environments.

Except for the laser range finder, the other sensors are connected to, and communicate with, the on-board computer using a *field bus* as shown in Fig. 2. 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.

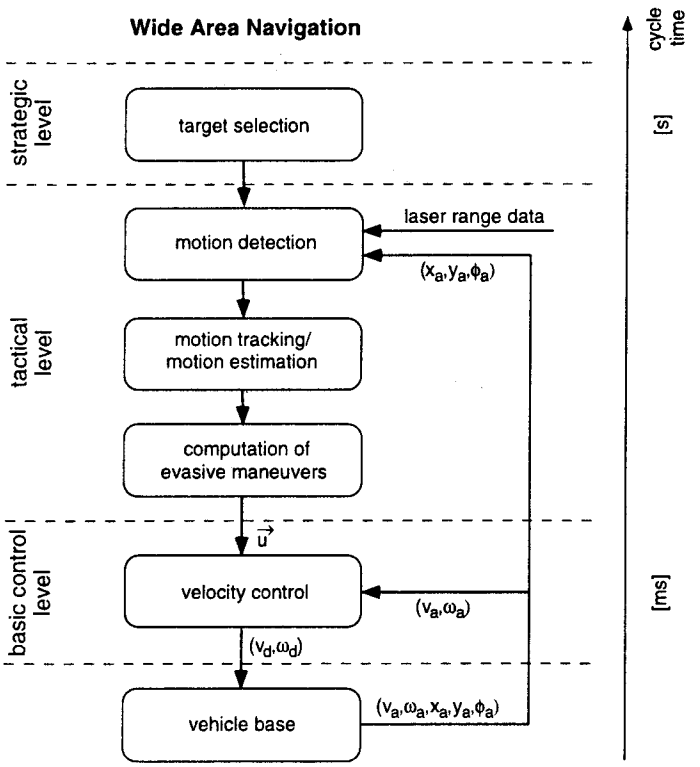

Figure 3: Software architecture of MAid's WAN module

The user interface of MAid's navigation system consists of the original wheelchair joystick and of a notebook computer. With the joystick the user points to the desired motion direction. The notebook is used to select MAid's operation mode and to enter the goal position. We are planning to enhance this interface with a commercial speech recognition system, similar to those currently used in the automobile industry. It is obvious however, that in order to be useful for a severely disabled person, MAid's user interface has to be adapted to that person's specific disability.

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 components of this control system which contribute to MAid's capability of navigating in a wide, crowded, rapidly changing area (WAN) are shown in Fig. 3. Note that in the following we simply denote these components as *WAN module.* For the sake of clearity, in Fig. 3 we omit the parts of the control system, which implement MAid's navigation skills for narrow, cluttered, partially unknown areas (NAN). These are described in [12].

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 sub-modules, a *motion detection* module, a module for *motion tracking and estimating object velocities and headings*, and a module for *computing evasive maneuvers.* In the following paragraphs we

give a brief description of the interaction of these submodules. The methods which they actually implement are described in detail in Section 5.

In order to be able to react to a rapidly changing environment with potentially many moving objects, MAid continuously observes the surrounding world with a 2D laser range-finder. The range data provided by this range-finder essentially represent MAid's view of the world. In the continuous stream of range data MAid tries to detect the objects in its environment and to identify which of these objects are stationary and which are in motion (see [13, 14] 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* [5] 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.

Motion detection, motion prediction, the computation of collision courses and the computaiton of the avoidance maneuver take approximately 70 ms. If we include the time for a sensor observation (recording of a range image) the cycle time increases to 0.3 sec, thus MAid is able to compute a new maneuver every 0.3 s. This is primarily due to the low transmission rate of the range-finder.

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 Rapidly Changing, Crowded Environments

In this section, we describe the methods and components which contribute to MAid's capability of navigating in a wide, crowded, rapidly changing area (WAN). Amongst existing robotic wheelchairs this capability is rather unique. The part of MAid's control system which implements this capability essentially consists of three components: an algorithm for motion detection, an algorithm for motion tracking, and an algorithm for computing evasive courses, which is based on the Velocity Obstacle (VO) approach [6].

5.1 Motion Detection and Motion 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.

5.1.1 Sensor selection and Estimation of Self-Motion

The sensor which we use for motion detection and tracking in crowded, rapidly changing environments is a laser range-finder (SICK PLS 200). In [13, 14] we argue that this is the most appropriate sensor for the given task. The device works on a time-of-flight principle. It has a maximum distance range of $d = 50\,\mathrm{m}$ with an accuracy of $\sigma_d \approx 50\,\mathrm{mm}$ and an angular range of $\pm 90\,\mathrm{deg}$ with a resolution of $0.5\,\mathrm{deg}$. The device is currently operated at a frequency of 3 Hz providing three range images per second.

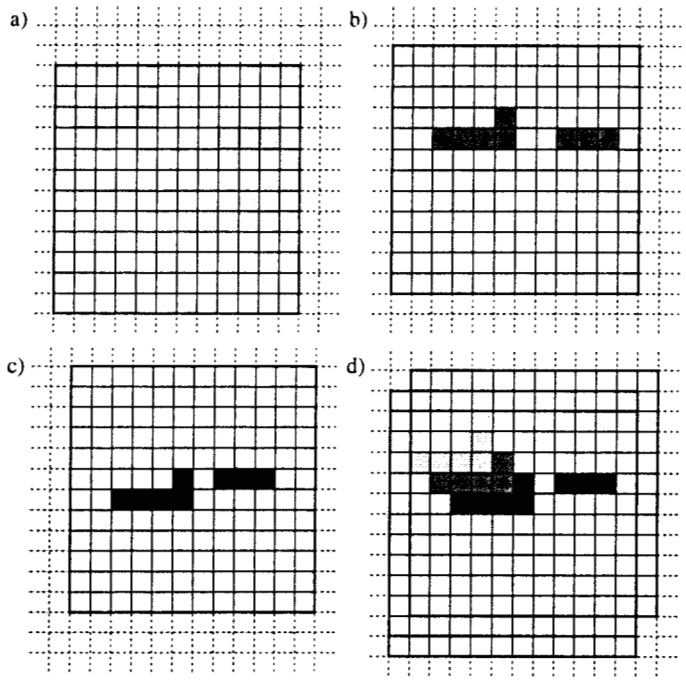

Figure 4: Time stamp maps: darker grey cells indicate a more recent observation.

The range data provided by the laser range-finder are naturally related to the local frame of reference attached to the sensor. In order to compare two subsequent local range images and to compute a differential image, it is necessary to know precisely which motion the sensor has undergone between two observations, how far it has moved from one viewpoint to the next and how far it has turned. This information is provided by the *dead-reckoning system*, which enables the wheelchair to keep track of its position and orientation over some limited travel distance with reasonable accuracy. With the information about the current position and orientation of the vehicle it is straightforward to transform the local range images from earlier measurements into the actual frame of reference.

5.1.2 Representations of Time-varying Environments

A very efficient and straightforward scheme for mapping range data is the *occupancy grid representation* [4]. 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.

Now, mapping large areas of free space is rather useless for detecting and tracking moving objects. To avoid this, 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

```
procedure detectMotion;
  for each cell ensemble cs_{x,t} describing an object x in TSM_t
    for each cell c_{i,t} in cs_{x,t}
      for each corresponding cell c_{i,t-1}, ..., c_{i,t-k}, ..., c_{i,t-n} in
            TSM_{t-1}, ..., TSM_{t-k}, ..., TSM_{t-n}
        if c_{i,t-k} carries a time stamp t - k
          then c_i is occupied by a stationary object
          else c_i is occupied by a moving object
  if majority of cells c_{i,t} in cs_{x,t} is moving
    then cell ensemble cs_{x,t} is moving
    else cell ensemble cs_{x,t} is stationary
```

Table 1: A motion detection algorithm based on a sequence of time stamp maps.

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, $TSM_{t-1}, \ldots, TSM_{t-n}$ of those time stamp maps. An example of such a sequence is shown in Fig. 4 a) - c). These pictures show three snapshots of a simple, time-varying environment with a moving and a stationary object in a time stamp map representation. The age of the observation is indicated through different gray levels where darker regions indicate more recent observations. Note that the maps are already aligned so that they have the same orientation. A translation by a corresponding position offset finally transforms the maps into the same frame of reference. The aligned maps are shown in Fig. 4-d). The assimilation of a range image into a 200 $\times$ 200 time stamp map takes 1.5 ms on a Pentium 166Mhz.

5.1.3 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. The algorithm which implements this idea is described in pseudo-code notation in Table 1.

As we pointed out earlier, the time stamp representation of a time-varying environment is more efficient for motion detection than commonly used grid representations. Particularly, the time stamp representation allows us to use a sequence of maps in a round robin mode without a need to clear and initialize the map which is used to assimilate the new sensor image. Outdated time stamps which originate from the mapping of previous sensor images do not have to be deleted but are simply overwritten. This procedure leaves the map receiving a new sensor image polluted by outdated information. However, this is not only efficient - as we save an expensive initialization operation - but is also correct. Cells which are marked by an outdated time stamp are simply considered as free space, which has the same effect as assigning some default value.

5.1.4 Motion Tracking and Estimation of Object Trajectories

Although kinematic and dynamic models of human walking mechanisms and gaits have been developed, 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. Therefore, the best we can

```
procedure findCorrespondence;
  for each object o_{i,t} in TSM_t
     for each object o_{j,t-1} in TSM_{t-1}
        CorrespondenceTable[i,j] = corresponding(o_{i,t}, o_{j,t-1});

function corresponding(o_{i,t}, o_{j,t-1});
  if o_{i,t} is stationary and o_{j,t-1} is stationary
     then ϑ = ϑ_s;  (threshold for stationary objects)
     else  ϑ = ϑ_m; (threshold for moving objects)
  if d(o_{i,t}, o_{j,t-1}) < ϑ
        and not_exists o_{k,t} :   d(o_{k,t}, o_{j,t-1}) < d(o_{i,t}, o_{j,t-1})
        and not_exists o_{l,t-1} : d(o_{i,t}, o_{l,t-1}) < d(o_{i,t}, o_{j,t-1})
     then return true;
     else  return false;
```

Table 2: An algorithm for tracking moving objects in a crowded environment.

do to track a moving object 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.

Note that while it is sufficient for motion detection to investigate only the mapping of two subsequent sensor images, provided the objects move at a sufficient speed, establishing a motion history may require to consider a more extended sequence of 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 two 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 step $t-1$. Obviously, this operation requires the objects to be represented in the same frame of reference.

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 images.

A description of the above algorithm in pseudo-code notation is given in Table 2. On a Pentium 166MHz a complete cycle involving both detecting and tracking any moving objects takes approximately 6 ms. For a more detailed description and discussion of our motion detection and tracking method we refer to [13, 14].

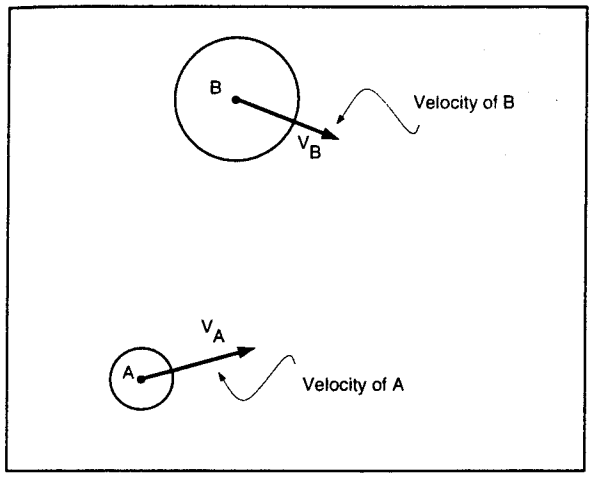

Figure 5: The mobile robot A and the moving obstacle B.

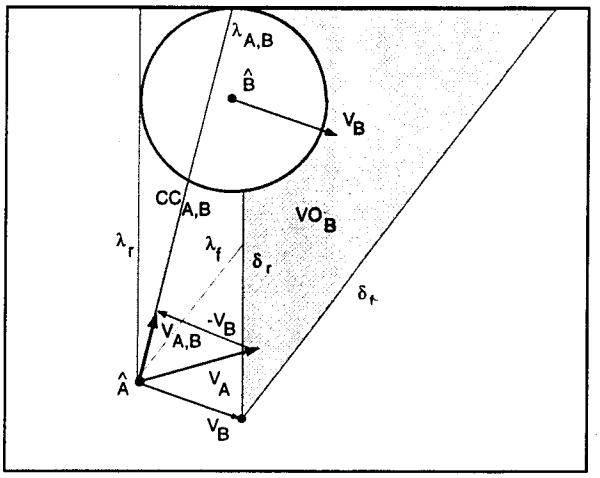

Figure 6: The Relative Velocity $\mathbf{v}_{A,B}$, the Collision Cone $CC_{A,B}$, and the Velocity Obstacle VO_B.

The algorithms in Table 2 allow us to establish a correspondence between the objects in two or even more subsequent time stamp maps. Having found this correspondence it is straightforward to compute estimates of the heading and the velocity of the objects. Let $cog(o_{i,t})$ and $cog(o_{i,t-1})$ be the centers of gravity of the visible contour of the object o_i at times t and $t-1$. Estimates for the velocity v and the heading ϕ of o_i are given by

$$v(o_{i,t}) = \frac{\| \vec{u}_{i,t} \|}{\Delta t} \quad \text{and} \quad \phi(o_{i,t}) = \text{atan}(\Im(\vec{u}_{i,t}), \Re(\vec{u}_{i,t})), \quad \text{where} \quad \vec{u}_{i,t} = cog(o_{i,t}) - cog(o_{i,t-1}).$$

As the above equations reveal, we use a very simple model for predicting the velocity and heading of an object. In particular, we assume that the object o_i moves linearly in the time interval from $t-1$ to t. Apparently, this may be a very coarse approximation of the true motion. The approximation, however, has proven to work sufficiently well at a cycle time of less than 100 ms for motion detection, motion estimation, and computation of an evasive course. At a slower cycle time a more sophisticated, nonlinear model for estimating the velocity and heading of an object may be appropriate.

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 5 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 6. Any relative velocity that

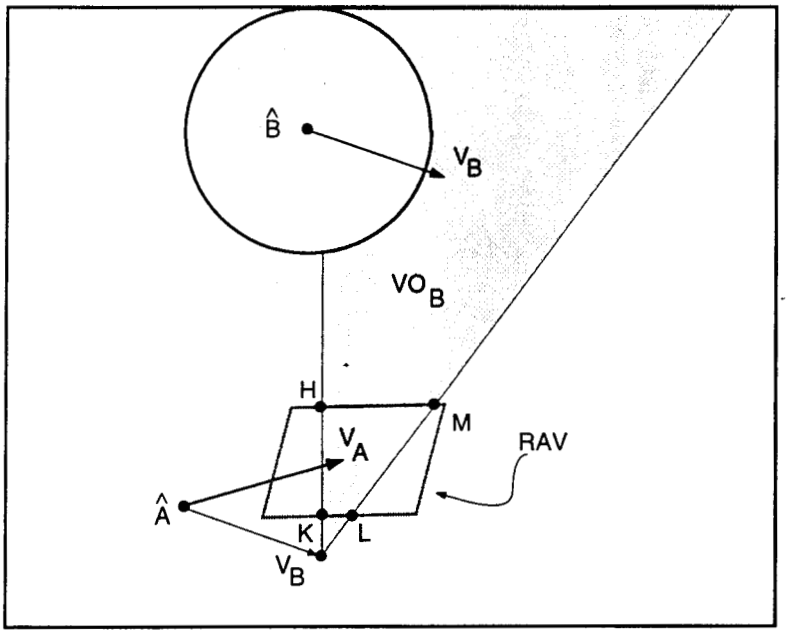

Figure 7: The reachable avoidance velocities RAV.

lies between the two tangents to $\widehat{B}$, 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 $\widehat{B}$ 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 6 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 $t < T_h$.

5.2.1 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 limited to those velocities that are physically reachable by robot A at a given state over a given interval. This set of *reachable velocities* is represented schematically by the polygon $KLMH$ shown in Figure 7. The set of *reachable avoidance velocities*, RAV, is defined as the difference between the reachable velocities and the velocity obstacle. A maneuver avoiding obstacle B can then be computed by selecting any velocity in RAV. Figure 7 shows schematically the set RAV consisting of two disjoint closed subsets. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

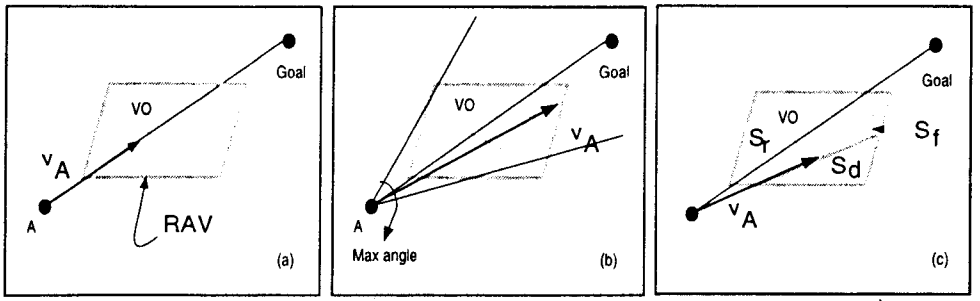

Figure 8: **a**: TG strategy. **b**: MV strategy. **c**: ST strategy.

It is possible then 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. For example, the only tangent velocities in Figure 7 are represented by the segments KH and LM of the reachable avoidance velocity set RAV. By choosing velocities in the set bound by segment HK or ML,, we ensure that the corresponding avoidance maneuver will avoid the obstacle from the rear, or the front, respectively.

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.

5.2.2 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. A global search have been proposed for off-line applications, and a heuristic search is most suitable for on-line navigation of the robotic wheelchair. The trajectory is generated 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 8. Selecting the highest feasible velocity in the general direction of the goal may reduce motion time, the MV heuristics shown in Figure 8. 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 8. 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.

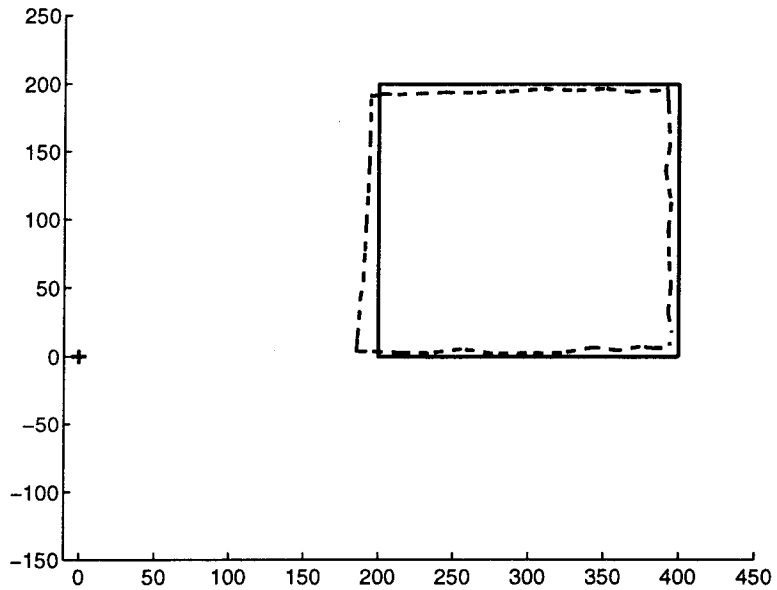

Figure 9: Tracking a single moving object with ground truth.

6 Experimental Results

MAid's performance was evaluated in two steps. Before taking the system to a realistic environment such as the concourse of a railway station, we conducted extensive tests under the simplified and controlled conditions of our laboratory. The laboratory embodiment of a "rapidly changing environment" consisted of an empty, delimited, and locked area where a second mobile robot moved on prescribed paths with known velocity profile, or groups of three and four people were asked to walk at moderate speed in front of MAid.

6.1 Experiments under laboratory conditions

To examine MAid's motion detection and tracking capability a commercial mobile robot Nomad XR4000 was programmed to move along a rectangular trajectory in front of the robotic wheelchair, equipped with the laser range-finder, at a distance of 2 m. The Nomad robot followed a velocity profile that made its center follow the polygonal trajectory represented by the solid line shown in Fig. 9. The wheelchair is identified by the cross mark at coordinates $(0, 0)$ in Fig. 9, and its position or orientation were not changed during the experiment.

The dotted line in the figure represents the motion of the Nomad robot as it was sensed and tracked by MAid. The estimated trajectory shows a maximum tracking error of less than 15 cm, which can possibly reduce the performance of the navigation algorithm. However, this is not the case, since the error is always reducing the estimate of the obstacle distance and therefore increases the navigation safety margins. The nature of the error can be easily understood by noticing that the trajectory estimation is carried out by tracking the center of the visible contour of mobile obstacle. Since the the visible obstacle is smaller than the true obstacle, its center will always result closer than its real position. Furthermore, the estimation error only affects the magnitude of the avoidance velocity and not its direction, which is computed using the left and right boundaries of the visible obstacle.

In a second set of experiments we asked a number of people to move at a comfortable walking speed along prescribed trajectories in an experimental area of approximately $4 \times 7\,\mathrm{m}^2$. The wheelchair with the range-finder was again kept stationary. The results of this set of experiments are shown in Fig. 10a) - d).

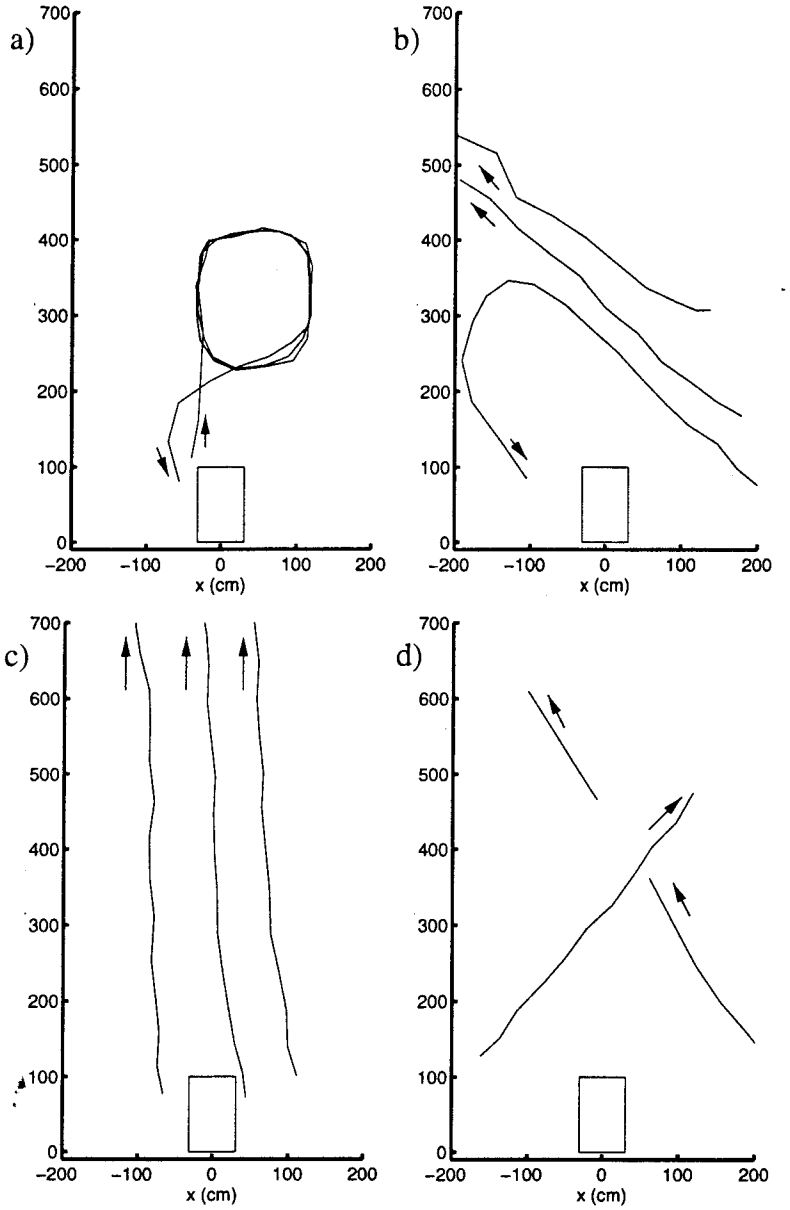

Figure 10: Tracking a group of people in a lab environment.

During the first experiment, a single person was asked to walk along a given rectangular trajectory in the area facing the range-finder sensor. After several laps, the person headed back to his initial position. The tracking algorithm tracked his motion in real time without any problem. The trajectory estimated by the tracking algorithm is shown in Fig. 10-a).

In the second experiment, three people moved across the field of view the wheelchair along straight lines, more or less parallel to each other, and the left most person made a sudden left turn and headed back to the wheelchair, as shown in Fig. 10-b). The subjects moved at slightly different speeds, so that their complete walk was visible by the range-finder. As shown in the figure, the tracking algorithm could easily track the motion of the three people. In the experiment shown in Fig. 10-c), we tracked the motion of three subjects moving again on parallel straight lines directly away from the wheelchair. This time the subjects moved at a similar speed.

The last experiment deserves a more detailed discussion. We let two subjects move along straight lines which crossed each other in front of the wheelchair. Accordingly, for a short period of time

one person was occluding the other from the range-finder view. Apparently then, the algorithm was unable to track the occluded person during this period of time. This loss of tracking manifests itself as the interruption of one of the trajectories, as shown in Fig. 10-d). Our algorithm lost the occluded person for two time steps. It detected and continued to track the motion after the subject became visible again.

Tracking moving objects whose trajectories cross each other is a very general problem and is not a specific problem of our tracking algorithm. Problems of this type cannot be eliminated even by very sophisticated methods such as those described in [2], which assume the knowledge of a model of the motion of the objects to be tracked. As mentioned above, we cannot make such an assumption since valid models of human motion are not available for our application domain. Experimental results further showing the performance of MAid's motion detection and tracking algorithm in a real environment are described in [13].

To complete the laboratory experiments, we evaluated the performance of the complete system, including motion detection, tracking and computation of avoidance maneuvers, under controlled conditions. This is a difficult task, since no metric is available to quantify the behavior of on-line algorithms reacting to unpredictable external events. Our experiment consisted of asking two subjects to approach the wheelchair at walking speed (approx. 1 m/s). The wheelchair's initial velocity was set to 0.5 m/s. The reaction of the system after it noticed the approaching objects is shown in a sequence of snapshots in Fig. 11. In the figure the wheelchair is represented by a rectangle, whereas the two subjects are represented by circles. The arrows attached to the rectangle and the circles represent the motion direction and velocity of MAid and the two people, respectively. The length of the each arrow represents to the distance traveled in one second. The entire experiment lasted less than five seconds as can be seen from the time stamps attached to the snapshots.

Before time 1.54 sec the two subjects moved in a safe direction without the danger of a collision. At time 1.54 sec one person changed direction and directly headed for the wheelchair. As we can observe, MAid reacted to this new situation by reducing its velocity and turning right. At time 3.1 sec the danger of a collision had disappeared again and MAid turned back to its initial direction and accelerated to its previous velocity. At time 4.14 sec one person had already left MAid's perceptual field when the other person suddenly made a turn and directly headed for MAid. Since this would have lead to an immediate collision MAid reduced its velocity to zero and stopped. Half a second later - the person had slowed down too and turned right a little - MAid accelerated again in a direction which allowed it to finally pass the person.

6.2 Experiments in the concourse of a railway station

After MAid had successfully passed a number of laboratory experiments similar those described above, the time had come to confront the real world. The real world was the concourse of the central station in Ulm, a hall of approximately $25 \times 40\,\mathrm{m}^2$. 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 11:00am and 1:30pm in fact, typically several tens of people stay and move around in the concourse, thus making it very suitable for the navigation experiments.

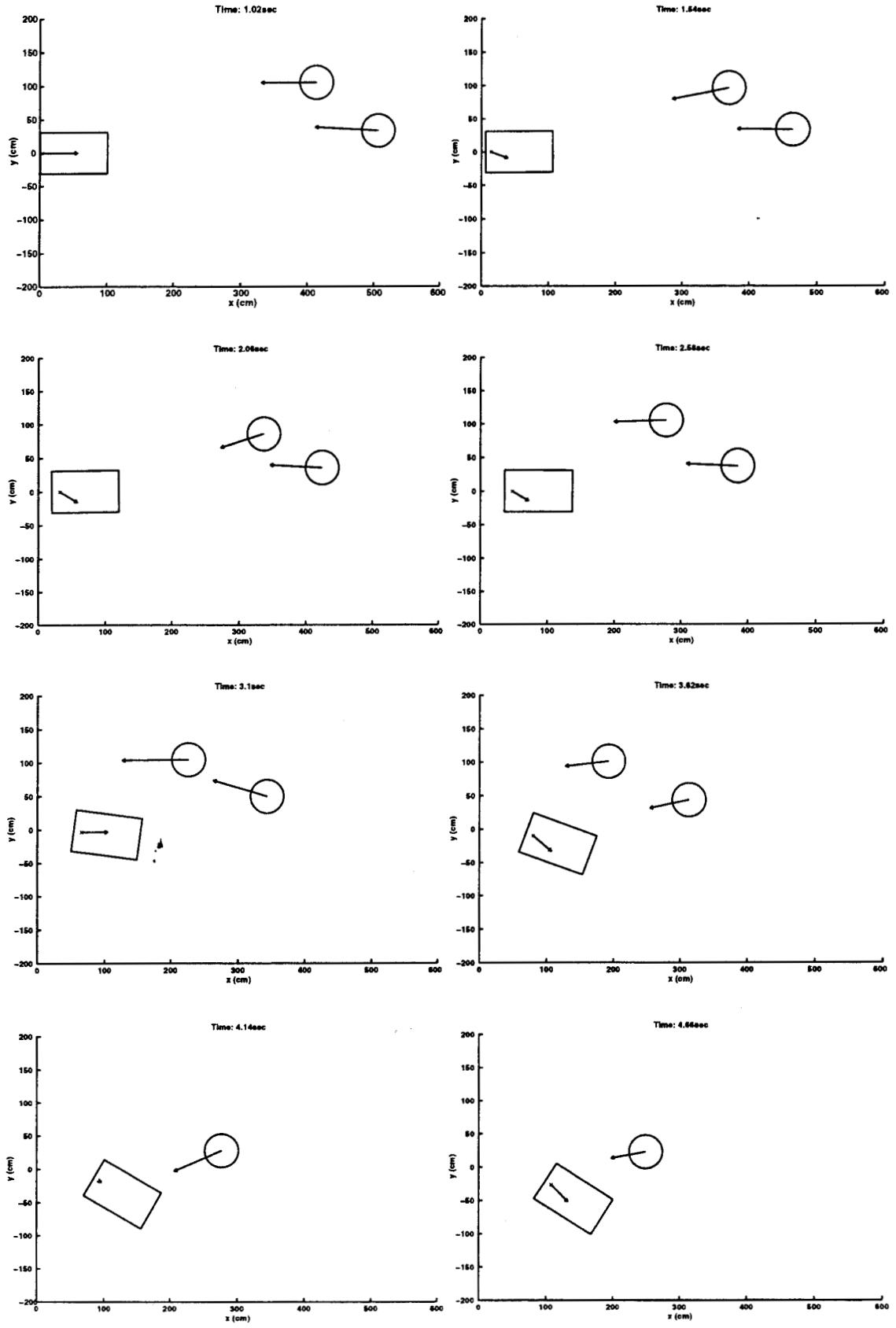

Figure 11: Lab experiment: MAid on a collision course with two approaching people.

Figure 12: MAid traveling in the concourse of a railway station.

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. 12. 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

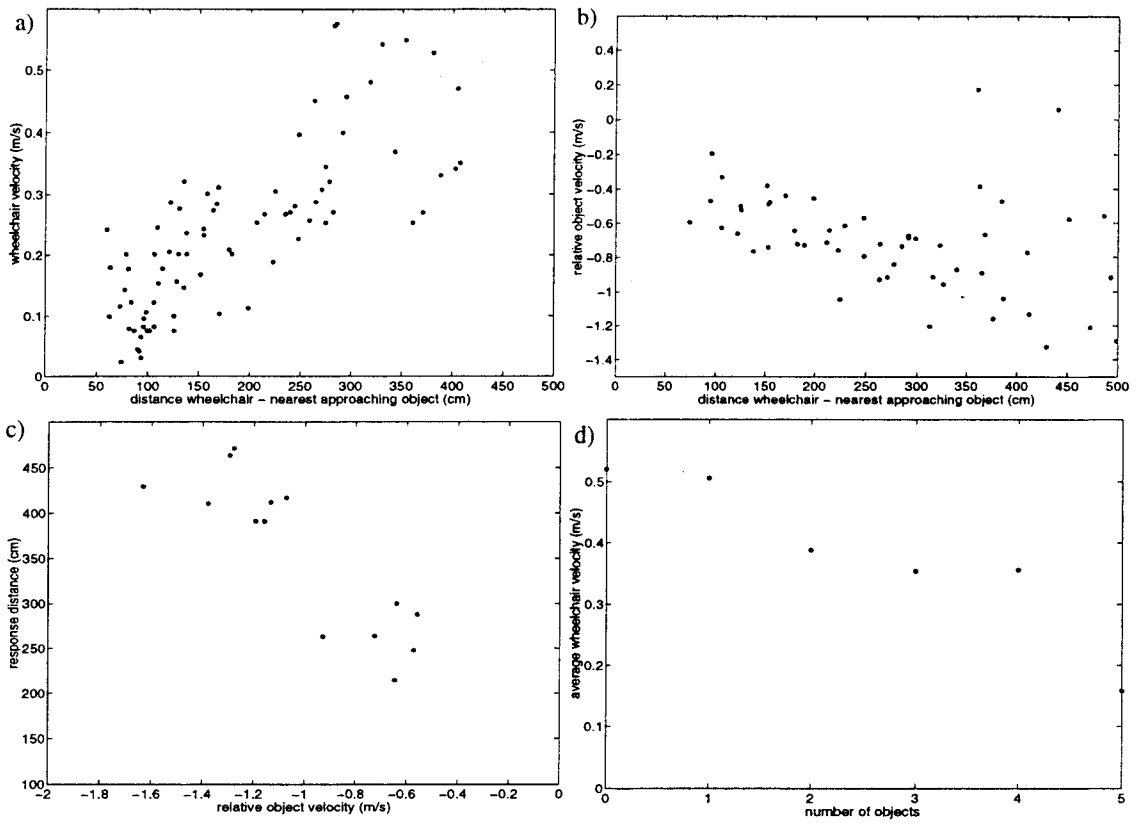

Figure 13: Illustration of MAid's performance in terms of wheelchair velocity and distance between the vehicle and the nearest object.

that situation: it waited until the group of people blocking its way had passed and then it continued its journey.

MAid's performance is demonstrated in the diagrams of Fig. 13, by showing the relations between some of the navigation variables such as, for example, wheelchair velocity, relative velocity between wheelchair and nearest objects, and clearance between the wheelchair and the nearest object. Fig. 13-a) shows the wheelchair velocity plotted over the distance between the wheelchair and the nearest object which is on a collision course with the wheelchair. The data in this diagram indicate that with decreasing clearance the wheelchair velocity drops to zero. Similarly, the velocity increases as the distance to the nearest approaching obstacle becomes larger. It is important to note that there is no unique causal relation between the wheelchair velocity and the clearance with the environment. Rather, the wheelchair's velocity depends on a number of quantities, such as the object motion direction, the object velocity, and the number of objects in MAid's proximity. This explains the variations in the data set. Note also that the distance to the nearest object is measured with respect to MAid's vertical axis and not with respect to the boundary of the wheelchair.

An equivalent dependency is shown in Fig. 13-b). There, the relative velocity between the wheelchair and the nearest object is plotted against the distance between the two objects. A negative value of the relative velocity means that the object is approaching the wheelchair, whereas a positive value means that the object is moving away from it. According to the data, the velocity of the nearest object relative to the wheelchair velocity decreases as the distance between the two objects decreases. This dependency describes the combined effect of motion planning and control algorithms, which reduces MAid's velocity whenever an object approaches the wheelchair. Note that this is not a unique causal correlation either.

In Fig. 13-c) we show the relation between the relative velocity of the nearest object and

the distance at which MAid starts an evasive maneuver. The larger the relative velocity of an approaching object, the sooner MAid initiates an avoidance maneuver, and the slower an object is approaching the wheelchair, the shorter is distance at which MAid starts to get out of its way. Fig. 13-d) finally shows MAid's average velocity plotted over the number of approaching objects. We can see that MAid decreases its speed the more objects are approaching on a collision course.

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 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 plan to 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.

7 Conclusion

in this paper, we presented the hardware and software design of the navigation system of our robotic wheelchair MAid. This navigation system enables MAid to move through crowded, rapidly changing environments, such as shopping malls and concourses of railway stations or airports, and also through narrow, cluttered, partially unknown environments. In this paper we only described the first of these two capabilities, which we denoted as WAN (wide area navigation). Three components essentially contribute to the capability to navigate in a wide, crowded, rapidly changing area: an algorithm for motion detection, an algorithm for motion tracking, prediction and the computation of potential collisions, and finally an algorithm for computing the avoidance maneuvers.

The algorithms for motion detection and tracking use the range data provided by a 2D laser range-finder. This sensor was chosen to facilitate the real-time capability of the tracking system. By using a laser range-finger our approach differs from the majority of known methods for motion detection and tracking which are based on visual information.

The time variation of the environment is captured by a sequence of temporal maps, which we call time stamp maps. A time stamp map is the projection of a range image onto a two-dimensional grid, whose cells coinciding with a specific range value are assigned a time stamp. Based on this representation we have discussed simple algorithms for motion detection and motion tracking, respectively. One complete cycle involving both motion detection and tracking takes approximately 6 ms. Our algorithms for motion detection and tracking do not presuppose the existence of kinematic and dynamic models of purposive human locomotion. Those models are not available in an environment such as a concourse of a railway station. With a cycle time of 6 ms for motion detection and tracking however, our approach is definitely "quick" and assures the required real-time capability.

The avoidance maneuvers are computed using the Velocity Obstacle approach, which allows the fast computation of the wheelchair velocity avoiding all static and moving obstacles. To take into account the environment uncertainty, an avoidance maneuver is computed at each sampling time, thus modifying in real time the nominal trajectory of the wheelchair. The complete trajectory to the goal is then computed incrementally, by selecting the avoidance velocities according to appropriate heuristics. The most commonly used heuristics has been to select an avoidance velocity in the general direction of the goal, to ensure that the wheelchair does not stray to far from its nominal trajectory, and can re-acquire its original goal after the obstacle avoidance.

MAid has undergone rather extensive testing. Its performance was tested in the central station of Ulm during rush-hour and in the exhibitions halls of the *Hannover Messe '98*, the biggest

industrial fair worldwide, during regular visiting hours. Altogether, MAid has 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, that can claim a similar performance.

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 has been 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] J. Barraquand, J.-C. Latombe. Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles. *Algorithmica*, 10, pp. 121-155, Springer-Verlag, 1993.

[2] Y. Bar-Shalom, T.E. Fortmann. *Tracking and Data Association.* Academic Press, 1987.

[3] 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.

[4] 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.

[5] 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.

[6] P. Fiorini, Z. Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *International Journal of Robotics Research*, July 1998, Vol. 17, No. 7, pp. 760-772.

[7] H. Hoyer, R. Hölper. Open Control Architecture for an Intelligent Omnidirectional Wheelchair. In *Proc. of the 1st TIDE Congress*, Brussels, pp. 93-97, IOS Press, 1993.

[8] 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.

[9] S. Mascaro, J. Spano, H. Asada. A Reconfigurable Holonomic Omnidirectional Mobile Bed with Unified Seating (RHOMBUS) for Bedridden Patients. In *In Proc of the 1997 IEEE Int. Conf. on Robotics and Automation*, Albuquerque, pp. 1277 - 1282, 1997.

[10] 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, pp. 203-224, 1995.

[11] D. Miller, M. Slack. Design and Testing of a Low-Cost Robotic Wheelchair Prototype. *Autonomous Robots*, 2, pp. 77-88, 1995.

[12] 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).

[13] E. Prassler, J. Scholz, M. Schuster, D. Schwammkrug. Tracking a Large Number of Moving Objects in a Crowded Environment. In *IEEE Workshop on Perception for Mobile Agents, Santa Barbara*, June 1998.

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

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

[16] P. Wellman, V. Krovi, V. Kumar. An Adaptive Mobility System for the Disabled. In *In Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, pp. 2006 - 2011, 1994.