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

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

Navigating a Robotic Wheelchair in a Railway Station during Rush Hour

Abstract

This paper describes the hardware design, control, and navigation system of and some preliminary experiments with the robotic wheelchair Mobility Aid for elderly and disabled people (MAid). MAid's general task is to transport people with severely impaired motion skills. The authors did not set out to reinvent and redevelop the set of standard skills of so-called intelligent wheelchairs, such as FollowWall, FollowCorridor, PassDoorway, which are commonly described in the literature. These maneuvers require motion control skills that disabled people, in spite of their disabilities, are eager to learn and quite good at using. Instead, this work focused on generalizing the approach to fine motion control by considering those maneuvers identified as very burdensome due to their duration and required concentration. 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 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 largest industrial fair in the world. Altogether, MAid has survived more than 36 h of testing in public, crowded environments with heavy passenger traffic.

1. Introduction

The freedom and capability to move around unrestricted and head for almost any arbitrary location is an extremely valuable commodity. Mobility has become an essential component of our quality of life. A natural consequence of this appreciation of good mobility is the negative rating of the loss of mobility caused, for example, by disease or 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 participating in a regular social life. Not infrequently, a loss of mobility leads to a loss of contact with other nondisabled people and makes it difficult to establish such contacts. A loss of mobility, whether due to an injury or to advanced age, is always accompanied by a loss of autonomy and self-determination, creates dependence, and in extreme cases may affect the individual intimacy and dignity.

The International Journal of Robotics Research
Vol. 18, No. 7, July 1999, pp. 711-727,
©1999 Sage Publications, Inc.

The loss of one's mobility may be seen as a difficult individual fate, but it is also a problem in society in general. The average age in Western societies is increasing dramatically. As a consequence, the number of people suffering from severe motion impairment will also increase. At the same time, we can observe an equally dramatic increase in the expenditures for health care and nursing and a reduction of nursing staff 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 incomes, and the elderly will be sent to nursing homes much earlier to receive sufficient care.

A way to avoid this unpleasant development may be through the development of robotic technologies. Many activities that 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, carrying, and moving around becomes feasible without the assistance of a nurse. People with motion impairment retain a certain amount of autonomy and independence and can remain in their familiar environment. Expenditures for nursing personnel or accommodation in a nursing home can be avoided or at least limited.

In this paper, we describe a robotic wheelchair Mobility Aid for elderly and disabled people (MAid) whose task is to transport people with severely impaired motion skills and 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 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 that most disabled people still want to execute themselves provided they have the necessary fine motor control.

Following this advice, we focused on developing an approach applicable to different types of motion skills and extending the autonomy of robotic wheelchairs. Our system has two modes of operation: a semiautonomous and a fully autonomous mode. In the semiautonomous 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 narrow area navigation (NAN). The implementation of this capability is described in Prassler, Scholz, and Strobel (1998).

In the second mode, MAid navigates fully autonomously through wide, rapidly changing, crowded areas, such as concourses, shopping malls, and convention centers. We denoted this latter type of motion skill as wide area navigation (WAN). 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 of by MAid. This application is particularly useful, since we were informed that, contrary to common expectation, people do not always give way to wheelchairs, as we also had occasion to experience during our experiments. This fact causes a lot of unnecessary fatigue, and automatic control in this situation would be quite welcome.

MAid's capability of navigating in rapidly changing environments was acknowledged as being useful and entertaining. MAid often had to work very hard to find its way through a crowd of people, and our test pilots were curious to see what MAid would do next (e.g., bump into a passenger, which it rarely 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 largest industrial fair in the world. Altogether, MAid has so far survived more than 36 h 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 that 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 wheelchairs. Quite the opposite is the case. In terms of its performance, WAN can be seen as a superset of functions such as AvoidObstacle 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 Section 2, 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 describe the software architecture and the algorithms that 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, which is presented in Prassler, Scholz, and Strobel (1998).

2. Related Work

Recently, there has been the development of a number of intelligent wheelchair systems. A design concept for a self-navigating wheelchair for disabled people was first proposed by Madarasz et al. (1986). That vehicle 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 Bell et al. (1994), 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 (1993) present a modular control architecture for an omnidirectional 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 a 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 Mascaro, Spano, and Asada (1997). RHOMBUS is a powered wheelchair with an omnidirectional drive that 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 the bed.

Mazo et al. (1995) describe an electrical wheelchair that 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 autonomous control mode. In this mode, the wheelchair follows a wall at a certain distance.

Miller and Slack (1995) 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 infrared 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.

Wellman, Krovi, and Kumar (1994) proposed a hybrid wheelchair that 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 an i860 coprocessor for the actuator coordination is used to control the wheelchair.

In recent years, much attention has been directed to motion planning in dynamic environments, which includes the problem addressed in this paper. Motion planning in dynamic environments is a difficult problem, since it requires planning in the state space; that is, simultaneously solving the path-planning and velocity-planning problems. It is a computationally difficult problem, and it is not guaranteed to have a solution due to the uncertain nature of the environment, since a solution computed at time t_0 may be infeasible at a later time (Sanborn and Hendler, 1988).

Motion planning in dynamic environments was originally addressed by adding the time dimension to the robot’s configuration space, assuming bounded velocities and known trajectories of the obstacles (Reif and Sharir, 1985; Erdmann and Lozano-Perez, 1987; Fujimura and Samet, 1989a). Another approach to dynamic motion planning is to decompose the problem into smaller subproblems: path planning and velocity planning. This method first computes a feasible path among the static obstacles. The velocity along the path is then selected to avoid the moving obstacles (Kant and Zucker, 1986; Lee and Lee, 1987; Fraichard, 1993; Fujimura and Samet, 1993).

A different approach consists of generating the accessibility graph of the environment, which is an extension of the visibility graph (Fujimura and Samet, 1989b). Online planning in dynamic environments has mostly emphasized reasoning and decision making, with little concern for robot dynamics (Sanborn and Hendler, 1988). These approaches rely on position information to test for collision between the robot and the moving obstacles, and thus can be described as zero-order methods, since they rely on position information to determine potential collisions.

A first-order method to compute the trajectories of a robot moving in a time-varying environment, using velocity information to directly determine potential collisions, is presented in Fiorini and Shiller (1997, 1998). The method uses the concept of velocity obstacle, which maps the dynamic environment into the robot velocity space. The velocity obstacle is a first-order approximation of the robot’s velocities that would cause a collision with an obstacle at some future time, within a given time horizon.

3. Hardware Design

MAid (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 and in crowded large-scale environments. The hardware core of the navigation system developed for the task is an industrial PC (Pentium 166 MHz) that 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 three segments each equipped with eight ultrasound transducers and a microcontroller 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 two-dimensional laser range finder PLS 200 mounted on a removable rack.

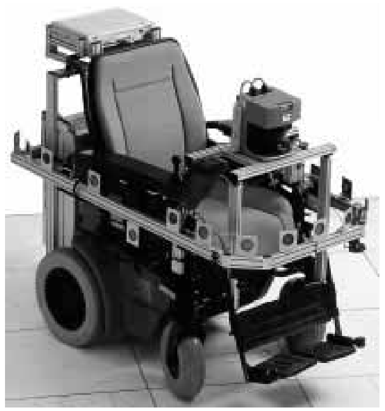
Fig. 1. The robotic wheelchair MAid.

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 used by MAid to perceive its surrounding environment. The sensors are used in combination, since one alone would not be sufficient to ensure a collision-free navigation in a natural environment with multiple moving obstacles.

The laser range finder provides an accurate two-dimensional picture of the environment because of its high angular and range resolution (see Section 5.1.1). However, a significant disadvantage of this device in a three-dimensional environment is that it covers only a single two-dimensional cross section of the environment at a given height. For this reason, we use the range information provided by the laser range finder mainly to track moving obstacles.

The sonar system is used to avoid collisions with objects that are not recognized by the laser range finder, either because they have a lower height and are out of reach of the laser beam (e.g., waste baskets in the concourse, bags and suitcases lying on the floor, or moving pets) or because they do not reflect the laser beam properly (e.g., glass doors). However, the sonar system has a rather poor angular resolution, which makes it impossible to track a larger number of moving obstacles. Altogether, the two sensors naturally complement each other in tracking and avoiding obstacles in a three-dimensional space.

Furthermore, data provided by sonar and laser systems are also used to estimate MAid’s position in the environment by means of an extended Kalman filter. An a priori description of the environment is required by the filter, which produces a set of expectations with regard to MAid’s sensor readings using a model of MAid’s locomotion and its last position data. 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.

Although all other components of MAid’s navigation system are inexpensive or can at least be substituted by cheaper components without reducing MAid’s performance, the laser range finder is undoubtedly an expensive sensor (approximately US$4000). However, as we have argued elsewhere (Prassler et al., 1998; Prassler, Scholz, and Elfes, 1999), no other sensor is equally suited for detecting and tracking a large number of moving objects in real time. This function in turn is essential for 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 Figure 2. The interface between these devices and the field bus is implemented by a number of microcontrollers (68HC11). Due to the high data rates of the laser range finder, this device is connected directly 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.

The user interface of MAid’s navigation system consists of the original wheelchair joystick and 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 to be useful for a severely disabled person, MAid’s user interface has to be adapted to that person’s specific disability.

One of the main criticisms that disabled people had when first confronted with MAid was its current appearance. It is obvious that MAid needs some major redesign to become an aesthetically pleasant device that does not attract people’s curiosity and exposes the user to unwelcome attention.

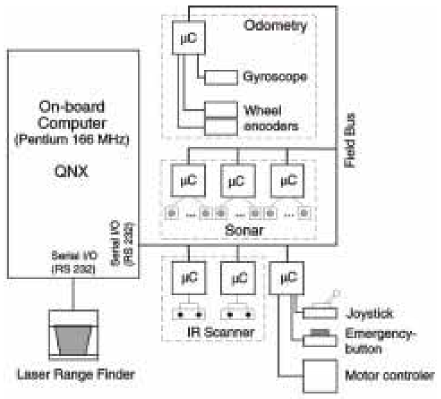

Fig. 2. Hardware architecture of MAid's control system.

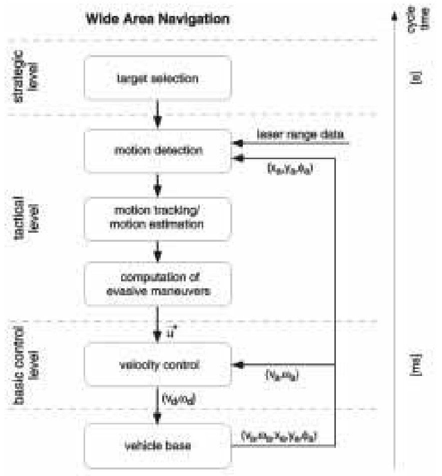

Fig. 3. Software architecture of MAid's wide area navigation module.

However, our discussion with the manufacturer of MAid's wheelchair about this issue revealed that a redesign was not considered a major problem. Most components of MAid's on-board sensing and electronics can be significantly reduced in size and placed in inconspicuous positions. The most difficult component to hide is the laser scanner, currently mounted on a removable aluminum frame on MAid's front, which has to be removed every time a person sits in the wheelchair. A preliminary analysis has shown that the laser scanner can still perform well when placed either in a less exposed position, such as an arm rest, or on a sliding or rotating boom, which would not interfere with the wheelchair access.

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 that contribute to MAid's capability of WAN (in a wide, crowded, rapidly changing area) are shown in Figure 3. Note that in the following, we simply denote these components as the WAN module. For the sake of clarity, in Figure 3 we omit the parts of the control system that implement MAid's NAN skills (for narrow, cluttered, partially unknown areas). These are described in Prassler, Scholz, and Strobel (1998).

On the basic control level, we compute the values of the control variables that 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 velocities. From these target values and the actual values provided by the vehicle's dead-reckoning system, the velocity controller computes appropriate correction values that 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 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 that they actually implement are described in detail in Section 5.

As mentioned previously, MAid uses its sonar system and its laser range finder to continuously monitor the surrounding environment. The range data provided by these sensors represent MAid's view of the world. In this continuous stream of range data, MAid tries to detect the objects in the environment and identify which are stationary and which are moving (for more details, see Prassler et al., 1998; Prassler, Scholz, and Elfes, 1999). Using laser range data, MAid further estimates the motion direction and velocity of the objects by extrapolating their past trajectories and velocities.

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 velocity obstacles (Fiorini and Shiller, 1993), 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 its vicinity.

Motion detection, motion prediction, the computation of collision courses, and the computation 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 s; thus, MAid is able to compute a new maneuver every 0.3 s. This is due primarily 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 plans that are more complex, 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, for example, a path planner, 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 that contribute to MAid's capability of navigating in a wide, crowded, rapidly changing area. Among existing robotic wheelchairs, this capability is rather unique. The part of MAid's control system that 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 approach (Fiorini and Shiller, 1998).

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 that we use for motion detection and tracking in crowded, rapidly changing environments is a laser range finder (SICK PLS 200). In Prassler et al. (1998) and Prassler, Scholz, and Elfes (1999), 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 $d = 50$ m, with an accuracy of $\sigma_d \approx 50$ mm and an angular range of $\pm 90°$, with a resolution of 0.5°. The device is currently operated at a frequency of 3 Hz providing three range images per second.

The range data provided by the laser range finder are naturally related to the local frame of reference attached to the sensor. 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

An efficient and straightforward scheme for mapping range data is the occupancy grid representation (Elfes, 1989). 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 mapping free space. Particularly, the farther away the observed objects were, the more time it took 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.

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; that is, 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 that 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 Figures 4a-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 observa-

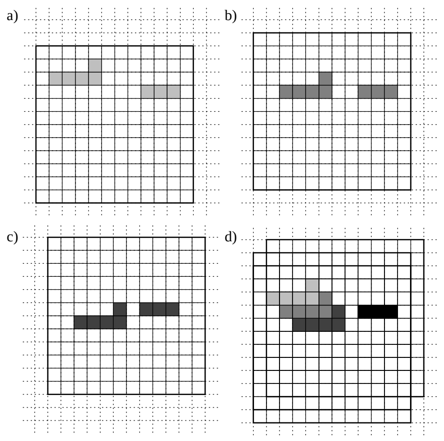

Fig. 4. A sequence of time stamp maps describing a toy environment. The age of the observation is indicated by different gray levels. The more recent the observation, the darker the gray level that marks the object contours.

Table 1. A Motion Detection Algorithm Based on a Sequence of Time Stamp Maps

```
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 $c_{i,t-1}, \dots, c_{i,t-k}, \dots,$
      $c_{i,t-n}$ in $TSM_{t-1}, \dots, TSM_{t-k}, \dots, TSM_{t-n}$
        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
```

tion is indicated by 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 Figure 4d. The assimilation of a range image into a 200×200 time stamp map takes 1.5 ms on a Pentium 166 Mhz.

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 that carries a time stamp t (occupied at time t) and test whether the corresponding cells in TSM_{t-1} were occupied too; that is, carried 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 that implements this idea is described in pseudocode notation in Table 1.

As we pointed out earlier, for motion detection, the time stamp representation of a time-varying environment is more efficient 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 that is used to assimilate the new sensor image. Outdated time stamps that originate from the mapping of previous sensor images do not have to be deleted but are simply overwritten. In this procedure, the map receiving a new sensor image remains polluted by outdated information. However, this is not only efficient—as we save an expensive initialization operation—but correct. Cells that 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 that would allow us to make inferences about the motion of a person over longer distances. Therefore, the best we can do to track a moving object in an 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 that describes the motion history of the moving objects from the spatial changes that we find in the mappings of these sensor images.

Note that although 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 one 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 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 the approximate distance that a person moving at fast walking speed covers between two sensor images.

A description of the above algorithm in pseudocode notation is given in Table 2. On a Pentium 166 MHz, a complete cycle involving both the detection 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 Prassler et al. (1998) and Prassler, Scholz, and Elfes (1999).

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 $(\text{cog}(o_{i,t}))$ and $(\text{cog}(o_{i,t-1}))$ be the centers of gravity of the object o_i as it is perceived by the laser range finder at times t and $t-1$. Estimates for the velocity v and the heading ϕ of o_i are given by

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

where

$$\vec{u}_{i,t} = cog(o_{i,t}) - cog(o_{i,t-1}).$$

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.

Table 2. An Algorithm for Tracking Moving Objects in a Crowded Environment

```
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 $\vartheta = \vartheta_s$;  (threshold for stationary objects)
    else $\vartheta = \vartheta_m$; (threshold for moving objects)
  if $d(o_{i,t}, o_{j,t-1}) < \vartheta$
      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;
```

5.2. Motion Planning Using Velocity Obstacles

In this section, we briefly summarize the concept of velocity obstacle for a single obstacle 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 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 let circle B represent an obstacle. To compute the velocity obstacle, 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 $CC_{A,B} = \{\mathbf{v_{A,B}} \mid \lambda_{\mathbf{A,B}} \cap \hat{\mathbf{B}} \neq \emptyset\}$, 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 gray 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 lies between the two tangents to $\hat{B}$, λ_f and λ_r, will cause a collision between A and

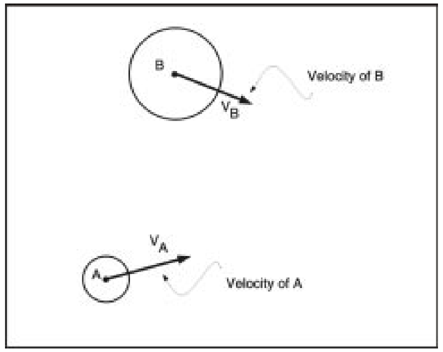

Fig. 5. The mobile robot A and the moving obstacle B.

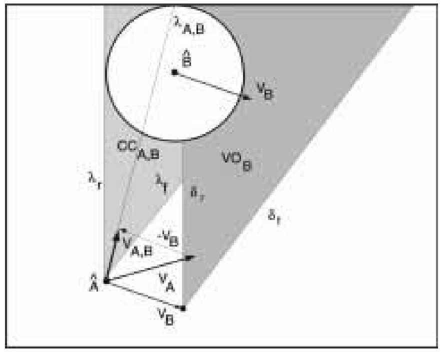

Fig. 6. The relative velocity $\mathbf{v_{A,B}}$, the collision cone $CC_{A,B}$, and the velocity obstacle VO_B.

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 by simply 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 gray sector. The velocity obstacle partitions the absolute velocities of A into avoiding and colliding velocities. Selecting $\mathbf{v_A}$ outside of the velocity obstacle would avoid collision with B. Velocities on the boundaries of the velocity obstacle 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 velocity obstacles.

In the case of many obstacles, obstacle avoidance is prioritized so that those with imminent collision will take precedence over those with a long time to collision. Furthermore, since the velocity obstacle 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 the RAV set. Figure 7 shows schematically the RAV set consisting of two disjoint closed subsets. For multiple obstacles, the RAV set may consist of multiple disjoint subsets.

It is then possible to choose the type of avoidance maneuver by selecting the 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

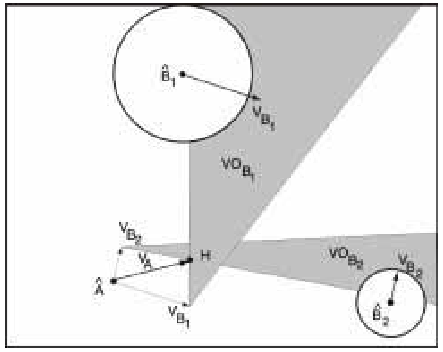

Fig. 7. The reachable avoidance velocities (RAV).

trajectories tangent to $\hat{B}$, 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 RAV set. 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 toward the goal, and satisfy the robot's dynamic constraints. A global search has 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 RAV set.

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 secondary goals. Choosing velocities in the RAV set (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 to-goal strategy shown in Fig. 8). Selecting the highest feasible velocity in the general direction of the goal may reduce motion time (the maximum velocity heuristics shown in Fig. 8). Selecting the velocity from the appropriate subset of RAV can ensure a desired trajectory structure (front or rear maneuvers) (the structure heuristics shown in Fig. 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 exists.

In the experiments described in the following section, we used a combination of the TG and ST heuristics to ensure that

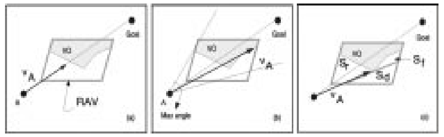

Fig. 8. (a) To-goal strategy, (b) maximum velocity strategy, (c) structure strategy.

the robotic wheelchair moves toward the goal specified by the user. When the RAV sets include velocity vectors aimed directly at the goal, the largest among them is chosen for the 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 heuristic adds 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 unmodeled uncertainties on the obstacle shapes and trajectories.

6. Experimental Results

MAid's performance was evaluated in two steps. Before taking the system to a real-world 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 profiles, 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 Figure 9. The wheelchair is identified by the cross mark at coordinates (0, 0) in Figure 9, and its position or orientation was not changed during the experiment.

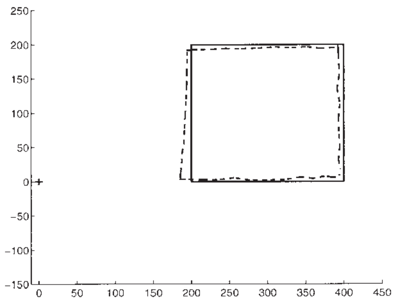

Fig. 9. Tracking a single moving object with ground truth.

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 contour of the mobile obstacle as it is perceived by the laser range finder. Since the visible part of the obstacle is smaller than the true obstacle, its center will always be closer than its real position. Furthermore, the estimation error affects only 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 4×7 m^2. The wheelchair with the range finder was again kept stationary. The results of this set of experiments are shown in Figures 10a-d.

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 Figure 10a.

In the second experiment, three people moved across the field of view of the wheelchair along straight lines, more or less parallel to each other, and the leftmost person made a sudden left turn and headed back to the wheelchair, as shown in Figure 10b. 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 Figure 10c, we tracked the motion of three subjects moving in 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 that crossed each other in front of the wheelchair. Accordingly, for a short period of time one person was occluding the other from the view of the range finder. Apparently, 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 Figure 10d. Our algorithm lost the occluded person for two time steps. It detected and continued to track the motion after the subject became visible.

Tracking moving objects whose trajectories cross each other is a very general problem and not specific to our tracking algorithm. Problems of this type cannot be eliminated even by very sophisticated methods such as those described in Bar-Shalom and Fortmann (1987), which assume the knowledge

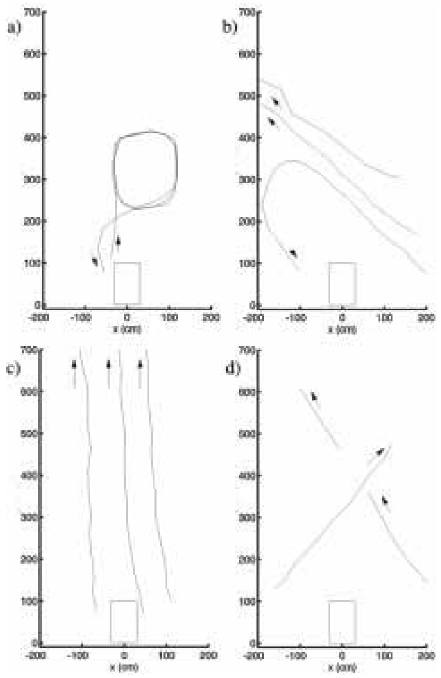

Fig. 10. Tracking a group of people in a lab environment.

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 showing the performance of MAid's motion detection and tracking algorithm in a real environment are described in Prassler et al. (1998).

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 (approximately 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 Figure 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 each arrow represents the distance traveled in 1 s. The entire experiment lasted less than 5 s, as can be seen from the time stamps attached to the snapshots.

Before 1.54 s, the two subjects moved in a safe direction without the possibility of a collision. At 1.54 s, one person changed direction and headed directly for the wheelchair. As we can observe, MAid reacted to this new situation by reducing its velocity and turning right. At 3.1 s, the possibility of a collision had disappeared, and MAid turned back to its initial direction and accelerated to its previous velocity. At 4.14 s, one person had already left MAid's perceptual field when the other person suddenly made a turn and headed directly for MAid. Since this would have led to an immediate collision, MAid reduced its velocity to zero and stopped. Half a second later—the person had slowed down as well and turned right a little—MAid accelerated again in a direction that 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 25×40 m^2. First test runs were conducted during the morning rush hour. 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 2 or 3 min. We counted up to 150 people crossing the concourse within about 1 min. After 2 or 3 min, however, the concourse was practically empty again, which did not leave us enough time to conduct 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 exhibits not the densest but the most continuous passenger traffic in the concourse. During the period between 11:00 a.m. and 1:30 p.m., 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 in the direction of the target location and by entering a travel distance. The wheelchair then starts moving in the desired direction as long as no collision is 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 Figure 12. The passenger traffic in these images is moderately dense, which actually facilitated the recording of the pictures. When the passenger traffic became too dense, MAid occasionally simply stopped, and did what a human operator probably would have done in that situation: it waited until the group of people blocking its way had passed and then continued its journey.

MAid's performance is demonstrated in the diagrams of Figure 13, which shows the relations between some of the navigation variables such as wheelchair velocity, relative velocity between wheelchair and nearest objects, and clearance between the wheelchair and the nearest object. Figure 13a shows the wheelchair velocity plotted over the distance between the wheelchair and the nearest object. 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 increases. It is important to note that there is no unique causal relation between the wheelchair velocity and the clearance between obstacles. Rather, the wheelchair's velocity depends on a number of factors, such as object motion direction, object velocity, and 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 the boundary of the wheelchair.

An equivalent dependency is shown in Figure 13b. 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.

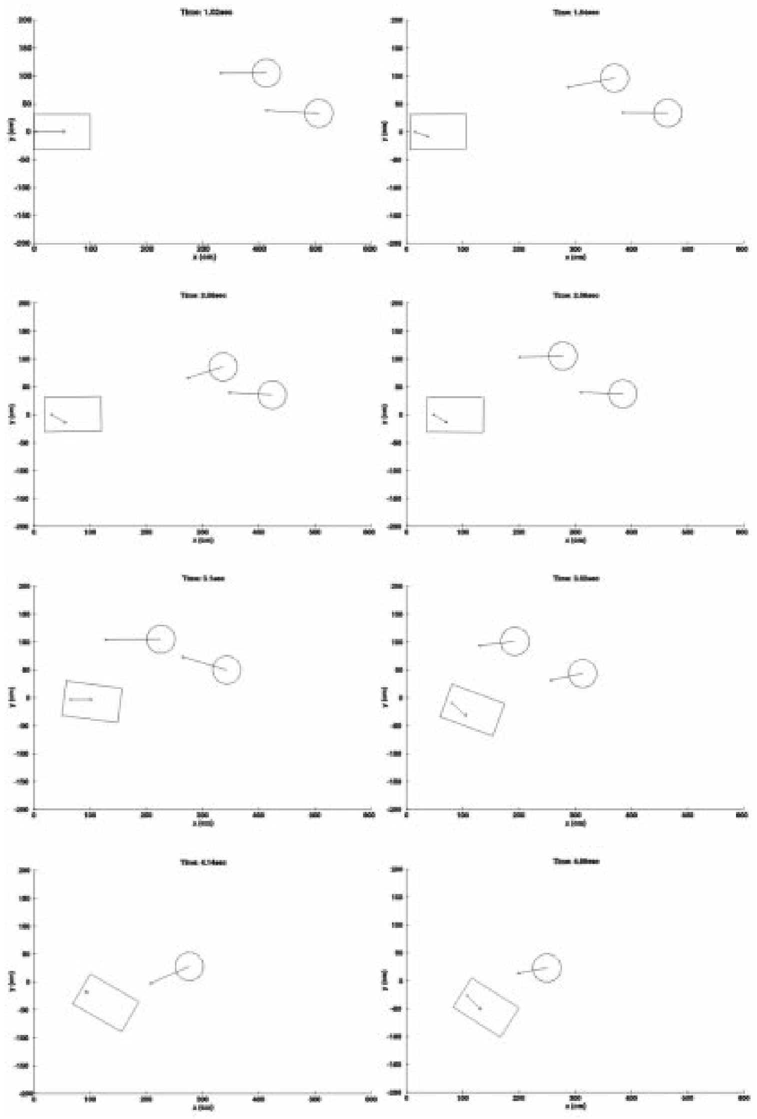
Fig. 11. Lab experiment: MAid on a collision course with two approaching people.

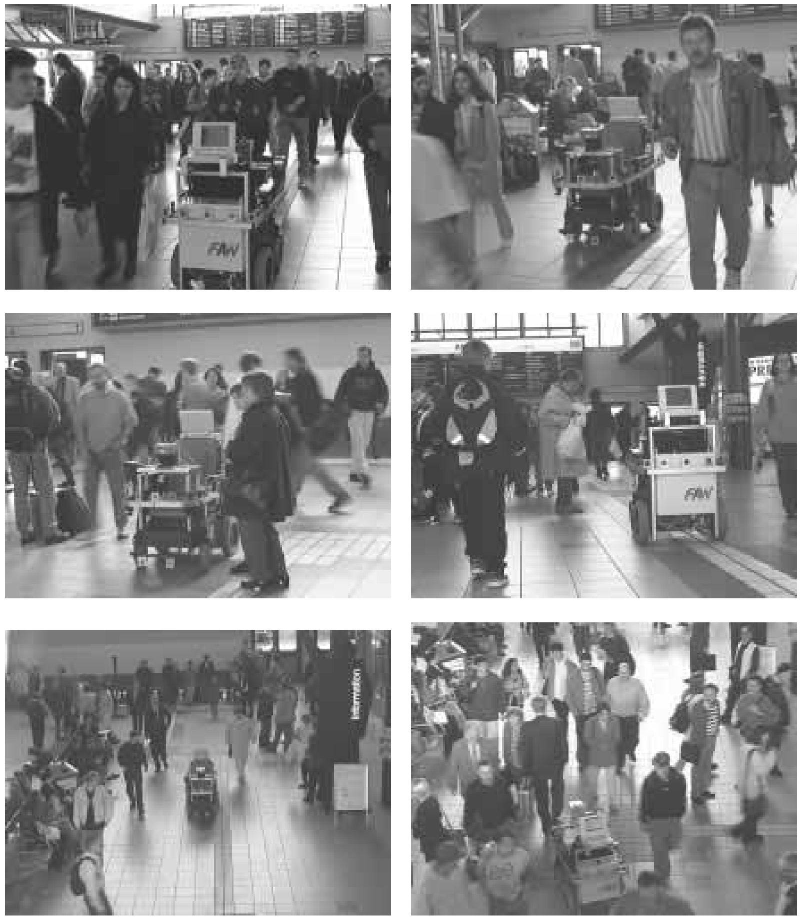
Fig. 12. MAid traveling in the concourse of a railway station.

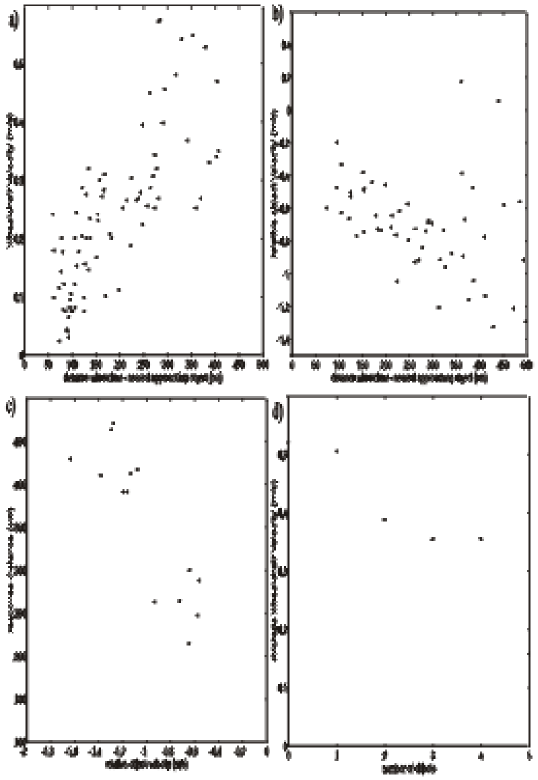

Fig. 13. Illustration of MAid's performance in terms of wheelchair velocity and distance between the vehicle and the nearest object.

In Figure 13c, 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, whereas the slower an object is approaching the wheelchair, the shorter is distance at which MAid starts to get out of its way. Figure 13d shows MAid's average velocity plotted over the number of approaching objects. We can see that MAid decreases its speed as the objects approaching on a collision course increase.

During our experiments in the concourse, MAid collided with objects several times. Usually, these objects were bags or suitcases lying on the floor, invisible to MAid's laser range finder and sonar sensors. To discover small obstacles in front of the wheelchair, we mounted two extra sonar sensors to the footrests of the wheelchair.

So far, MAid has survived about 18 h 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 Messe '98. The Hannover Messe is the largest industrial fair in the world. In Hannover, MAid drove through the exhibition halls for seven days between 2 and 3 h per day at regular visiting hours. Altogether, MAid has successfully navigated in crowded, rapidly changing environments for more than 36 h.

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 through narrow, cluttered, partially unknown environments. In this paper, we described only the first of these two capabilities, which we denoted as WAN. 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 an algorithm for computing the avoidance maneuvers.

The algorithms for motion detection and tracking use the range data provided by a two-dimensional 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 ensures 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 heuristic has been to select an avoidance velocity in the general direction of the goal and to ensure that the wheelchair does not stray too far from its nominal trajectory and can reacquire its original goal after obstacle avoidance.

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

Bar-Shalom, Y., and Fortmann, T. E. 1987. *Tracking and Data Association*. New York: Academic Press.

Bell, D. A., Borenstein, J., Levine, S. P., Koren, Y., and Jaros, L. 1994. An Assistive navigation system for wheelchairs based upon mobile robot obstacle avoidance. *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego.

Elfes, A. 1989. *Occupancy grids: A probabilistic framework for robot perception and navigation*. Ph.D. thesis, Electrical and Computer Engineering Department/Robotics Institute, Carnegie-Mellon University.

Erdmann, M., and Lozano-Perez, T. 1987. On multiple moving objects. *Algorithmica* 2:477–521.

Fiorini, P., and Shiller, Z. 1993. Motion planning in dynamic environments using the relative velocity paradigm. *Proc. of 1993 IEEE Int. Conf. on Robotics and Automation*, Atlanta.

Fiorini, P., and Shiller, Z. 1997. Time optimal trajectory planning in dynamic environments. *J. Appl. Math. Computer Sci.* 7(2):101–126.

Fiorini, P., and Shiller, Z. 1998. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robotics Res.* 17(7):760–772.

Fraichard, T. 1993. Dynamic trajectory planning with dynamic constraints: A state-time space approach. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Yokohama, Japan, pp. 1393–1400.

Fujimura, K., and Samet, H. 1989a. A hierarchical strategy for path planning among moving obstacles. *IEEE Trans. Robotics Automation* 5(1):61–69.

Fujimura, K., and Samet, H. 1989b. Time-minimal paths among moving obstacles. *Proc. of 1989 IEEE Int. Conf. on Robotics and Automation*, Scottsdale, pp. 1110–1115.

Fujimura, K., and Samet, H. 1993. Planning a time-minimal motion among moving obstacles. *Algorithmica* 10:41–63.

Hoyer, H., and Hölper, R. 1993. Open control architecture for an intelligent omnidirectional wheelchair. In *Proc. of 1st TIDE Congress*. Amsterdam: IOS Press, pp. 93–97.

Kant, K., and Zucker, S. W. 1986. Towards efficient trajectory planning: The path-velocity decomposition. *Int. J. Robotics Res.* 5(3):72–89.

Lee, B. H., and Lee, C.S.G. 1987. Collision-free motion planning of two robots. *IEEE Trans. Sys. Man Cyber.* SMC-17(1):21–32.

Madarasz, R. L., Heiny, L. C., Cromp, R. F., and Mazur, N. M. 1986. The design of an autonomous vehicle for the disabled. IEEE J. Robotics Automation RA-2(3):117–126.

Mascaro, S., Spano, J., and Asada, H. 1997. A reconfigurable holonomic omnidirectional mobile bed with unified seating (RHOMBUS) for bedridden patients. *Proc. of 1997 IEEE Int. Conf. on Robotics and Automation*, Albuquerque, pp. 1277–1282.

Mazo, M., Rodriguez, F. J., Lazaro, J. L., Urena, J., Garcia, J. C., Santiso, E., Revenga, P. A., and Garcia, J. J. 1995. Wheelchair for physically disabled people with voice, ultrasonic and infrared sensor control. *Autonomous Robots* 2:203–224.

Miller, D., and Slack, M. 1995. Design and testing of a low-cost robotic wheelchair prototype. *Autonomous Robots* 2:77–88.

Prassler, E., Scholz, J., and Elfes, E. 1999. Tracking people in a railway station during rush-hour. *Proc. of 1st Int. Conf. on Computer Vision Systems ICVS'99*, Gran Canaria, Spain, pp. 162–179.

Prassler, E., Scholz, J., Schuster, M., and Schwammkrug, D. 1998. Tracking a large number of moving objects in a crowded environment. *IEEE Workshop on Perception for Mobile Agents*, Santa Barbara.

Prassler, E., Scholz, J., and Strobel, M. 1998. MAid: Mobility assistance for elderly and disabled people. *Proc. of 24th Int. Conf. of the IEEE Industrial Electronics Soc. IECON'98*, Aachen, Germany.

Reif, J., and Sharir, M. 1985. Motion planning in the presence of moving obstacles. *Proc. of 25th IEEE Symp. on the Foundation of Computer Science*, pp. 144–153.

Sanborn, J. C., and Hendler, J. A. 1988. A model of reaction for planning in dynamic environments. *Int. J. Artif. Intell. Eng.* 3(2):95–101.

Wellman, P., Krovi, V., and Kumar, V. 1994. An adaptive mobility system for the disabled. *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, pp. 2006–2011.