ENSURING SAFE OBSTACLE AVOIDANCE IN A SHARED-CONTROL SYSTEM

Thomas Röfer and Axel Lankenau

Bremer Institut für Sichere Systeme, TZI, FB 3 Mathematik/Informatik, Universität Bremen,
Postfach 330440, 28334 Bremen, Germany, T.+49 421 2184659, F.+49 421 2183054, {roefer,alone}@tzi.de

Abstract *– This paper describes the Bremen Autonomous Wheelchair project. It shows that common static fire strategies for ultrasonic sensors are inherently unsafe and proposes a new adaptive fire strategy which delivers a complete coverage of the environment of the robot. Furthermore, a new obstacle avoidance approach for shared-control robotics systems is described in detail.*

1. INTRODUCTION

The *Bremen Autonomous Wheelchair* project covers a variety of research issues that include *spatial cognition*, *safe systems*, and *rehabilitation*. Emphasizing one or the other of these cornerstones provides various views of the wheelchair *Rolland* (cf. Fig. 1). Within the framework of the priority program "Spatial Cognition" of the Deutsche Forschungsgemeinschaft, the main topics are navigation, landmark recognition and space representation (cf. [10]). Moreover, the wheelchair serves as a case study for the development of safe systems based on formal methods (cf. [11]). The third determinant of the Rolland project is the aspect of rehabilitation (cf. [14]), i. e. to develop a support tool that can help handicapped and elder people to control a wheelchair. These seemingly divergent topics complement each other to build a robust and safe wheelchair system.

Rolland is based on the commercial power wheelchair *Genius 1.522* manufactured by the German company Meyra. The wheelchair is a non-holonomic vehicle that is driven by its front axle and steered by its rear axle. The human operator can control the system with a joystick. The wheelchair is equipped with a standard PC (Pentium 233, 64 MB RAM) and sensors to perceive its environment.

The principal idea behind such an assistive system is to wiretap the connection between the joystick and the motor. If the human operator issues a command that may lead to a collision with an obstacle, the wheelchair changes the dangerous target command to a safe one.

This paper shows how the tasks of detecting obstacles in the environment and avoiding these obstacles are solved in the Bremen Autonomous Wheelchair. It especially deals with the problems that arise due to the fact that, in contrast to most research robots, the wheelchair is a shared-control system, i. e. it is jointly controlled by the assistive system and the human operator.

Figure 1: The Bremen Autonomous Wheelchair "Rolland" *is a standard product enhanced with sonar sensors and a PC.*

2. SONAR MEASUREMENTS

Beside laser range finders, ultrasound sensors are the most common sensors for measuring ranges in mobile robotics. They are much cheaper and smaller than laser sensors, so a lot of them can be mounted around a mobile robot, in contrast to only one or two laser scanners, as can be found on many experimental mobile systems, e. g. [8, 5, 9]. As a wheelchair has a lengthy shape, many laser range finders would be necessary to completely perceive the vehicle's environment. Therefore, Rolland was equipped with 27 Polaroid sonar sensors that are fixed around the wheelchair at a height of 50 cm.

Sonar sensors measure distances by sending out an

ultrasound pulse [2]. This pulse is reflected back to the sensor by the objects in the environment. Thus, the sensor can measure the flight time of the sonar pulse until the first reflection returns. From this period of time, the distance to the closest object can be determined. Starting at the sensor, the sonar pulse spreads along a cone with a certain angle, i. e. the *opening angle* of the sensor. Thus, sonars have a high range resolution, but a poor angular resolution, because they only measure the distance to a single object in their opening cone.
In addition to this low precision for larger distances, there are several reasons for possible misreadings of ultrasound sensors:

2.1 Specular Reflections

It is possible that an object does not reflect enough energy of the sonar pulse back to the sensor to be noticed. This is either the case if an object is very small or if the sonar pulse hits its surface under a small, non-perpendicular angle. As a result, the distances measured are too large, i. e. obstacles are overlooked. To reduce the risk of missing an object by measuring it under an inconvenient angle, the ultrasonic sensors are fixed to the wheelchair's front and back in circular arrangements. Thus, at least one sensor is oriented in a nearly perpendicular angle towards potential obstacles. However, it is still impossible to detect very convex corners as well as very small objects. Therefore, it is a pre-requisite for the use of Rolland that such objects do not exist in the wheelchair's area of operations.

2.2 Cross-Talks

In most sonar range measuring systems, a sensor cannot determine whether the pulse it receives was sent out by itself or by another sensor. In addition, it is even possible that a pulse of an earlier measurement cycle reaches the sensor. In both cases, the distances measured are too small, because the disturbing pulses have reached the sensor earlier than its own one. To reduce cross-talks, only a few sensors are fired at the same time, and their orientations should be as different as possible. Some sonar measuring systems were even designed in a way that they can only take a few readings simultaneously, e. g. the Nomadic *Sensus 100* system used in Rolland can only fire two sensors at the same time, one to the left side and one to the right side. Therefore, sonar readings from different sensors are taken in a sequence, i. e. the so-called *fire strategy.* To reduce cross-talks over different measuring cycles, even the orientation of sensors fired one after the other should differ as much as possible. Hence, a fire strategy activates the sensors in a pseudo-random, alternating sequence.

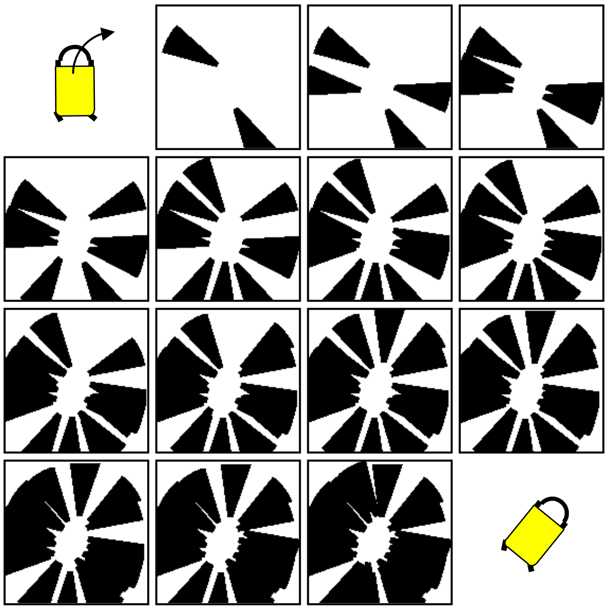

Figure 2: A static sonar fire strategy. Black color marks the areas measured by the sonar sensors during a complete firing cycle of 14 steps while the wheelchair drives a narrow right curve.

Figure 2 visualizes such a fire strategy as well as a potential problem of this approach. The wheelchair is equipped with 27 ultrasound sensors, out of which it can simultaneously fire two. Thus, due to the sensor arrangement, the wheelchair would acquire a continuous scan of the environment after 14 sonar measurement cycles if it did not move meanwhile. However, things change when the wheelchair is moving. If, e. g., the vehicle drives a narrow curve to the right, as depicted in Fig. 2, the readings give only a holey image of the surroundings, because the positions where readings were taken are shifted by the wheelchair's motion. Thus, there is a risk of overlooking obstacles when using such a static fire strategy, not only for the Bremen Autonomous Wheelchair, but also for other mobile robots with sonar sensors, e. g. the *Nomad 200* manufactured by Nomadic (used e. g. in [13]) or RWI's *B21* (used e. g. in [4]). Section 3.3 will present a dynamic fire strategy that solves this problem.
To further reduce the impact of cross-talks on the wheelchair's behavior, only obstacles are taken into account that were measured at least twice. The dynamic fire strategy that will be presented in section 3.3 ensures that the demand for two registrations of an obstacle does not significantly slow down the detection process.

2.3 Blindness in the Sensor's Immediate Vicinity

Most ultrasound sensors use only a single membrane to both generate the sonar pulse and measure it when it returns. Therefore, the membrane needs a certain time period after sending out the pulse to become calm, so that the vibration resulting from the pulse's return can be registered. During this period of time, no incoming sonar signals are noticed. This results in a blindness for objects that are very close to the sensor, e. g., the sonars of the Bremen Autonomous Wheelchair cannot measure ranges of less than 15 cm.

3. STOPPING IN TIME

One of the major tasks of the Bremen Autonomous Wheelchair is to avoid collisions. On the lowest level, the safety level, this means *stopping in time* before a collision can occur. On higher levels (cf. section 4), this goal can also be reached by driving around obstacles.
To be able to determine when it is necessary to brake the wheelchair, several things must be taken into account:

- The wheelchair's deceleration and its reaction time. Together with the actual speed, these values determine the vehicle's overall stopping distance.
- The *travel distance* to the closest obstacle. This is the distance the wheelchair can travel with its current driving direction and its actual steering angle before it hits the first obstacle.

Whereas it is a simple operation to calculate the overall stopping distance for a certain speed, the travel distance to the closest obstacle cannot be determined straightforward. The wheelchair's motion is non-holonomic and its body has a complex shape. In addition, the sensors do not give complete snapshots of the environment, instead, they deliver their readings over a longer period of time (approximately 0.5 s) while the wheelchair moves, i. e. the vehicle could cover a distance of 42 cm before all sonars are fired once. Therefore, the problem of calculating the travel distance to the closest obstacle was separated into two sub-problems: the generation of a *local obstacle map*, and the calculation of the *distance to the closest obstacle* from the current state of the map using *virtual sensors*, assuming that all relevant obstacles were already measured.

3.1 Local Obstacle Map

The local obstacle map or *occupancy grid* [6] is a quadratic array of cells, each of which encodes—among other things (cf. section 3.3)—the presence of an obstacle at the corresponding position in the environment of the wheelchair. Due to computational considerations, the position of the wheelchair relative to the map is static with respect to translational movements and dynamic with respect to rotational changes. This means, that the wheelchair's (x/y)-position, i. e. the middle of the front axle, constantly remains in the center of the map, whereas the orientation of the wheelchair relative to the map changes during operation. As a consequence of the wheelchair's static position at the map center, the representation of the environment, i. e. the positions of obstacles, has to be shifted according to the current movement of the vehicle that can be determined from the wheelchair's dead reckoning system. Since the map is represented as an array in memory, the shifts in x and y direction are computationally cheap. In contrast to that, a rotation of the array would be rather time consuming and prone to discretization errors. Therefore, instead of rotating the map, the wheelchair's current rotational deviation to the map is continuously determined. As the map is relatively small in comparison to the wheelchair's speed, odometry errors do not have a negative impact on the consistency of the readings in the map, because they will leave the map shortly after they were measured.
To be able to stop in time using the map, its extent from the center to any side must be at least the sum of the following distances:

- The wheelchair's extent from the map's center. When driving forwards, this is the wheelchair's length in front of the driving axle, i. e. 40 cm, otherwise it is the length of the back, i. e. 80 cm.
- A minimum range between the obstacles and the wheelchair. As the sonar sensors cannot measure ranges shorter than 15 cm, the vehicle will never come closer to objects than this distance.
- The overall stopping distance. For the wheelchair's maximum speed of 84 cm/s, this is 53 cm. In backward direction, the vehicle can only travel up to 42 cm/s, so the stopping distance is shorter, i. e. 18 cm.
- The distance the wheelchair can cover until all relevant readings have been inserted into the map. In the worst case, all sensors must have been fired, which takes approximately 0.5 s. Again, it must be distinguished between forward motion (42 cm) and backward motion (24 cm).

So, although the wheelchair's back is longer than its front, the required map size is maximal for forward motion (2×150 cm), because the vehicle's maximum speed is much smaller in backward direction, and thus the backward overall stopping distance (2×137 cm) is significantly smaller, too. As the map is also used for another application that smoothes the acceleration and deceleration of the wheelchair, its size is larger: 402×402 cm^2. In order to be sufficiently precise to allow, e. g., driving through doorframes, the map consists of 17956 3×3 cm^2 cells.

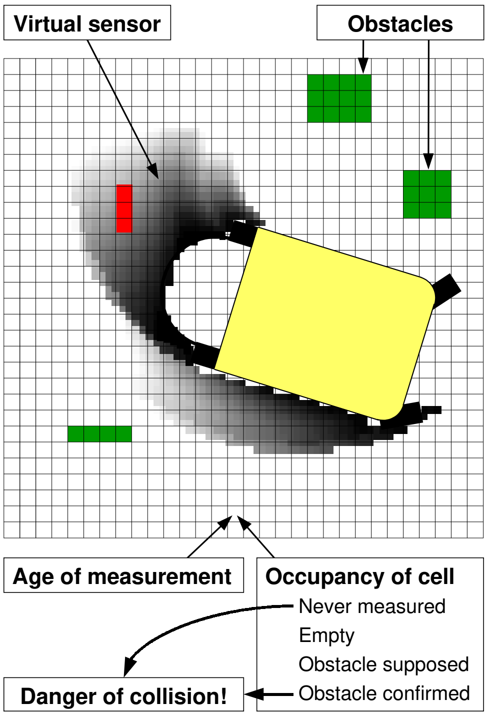

Figure 3: The local obstacle map and a virtual sensor. The virtual sensor comprises the cells the wheelchair will "visit" when driving with the steering angle indicated by the back wheels. The darker the cells, the earlier this contact will happen.

3.2 Distance to Closest Obstacle

Before it will be discussed how measurements are entered into the map, the determination of the travel distance to the closest obstacle is presented. It is based on the assumption that all relevant obstacles are already represented in the map. In order to decide whether such a representation of obstacles may potentially cause a collision with the wheelchair, the vehicle's trajectory must be anticipated. This is a rather complex task for a wheelchair with a driving axle and a steering axle, because parts of the body swing out sideways when driving curves. Steering even changes the shape of the system's body, because the turned wheels can stick out sideways. In contrast to other approaches that try to reduce the computational costs of the anticipation of the trajectory by simplifying the shape of the region that has to be searched for obstacles (cf. e. g. [7]), the Bremen Autonomous Wheelchair uses an exact method: the virtual sensors (cf. Fig. 3).

Virtual Sensors

For each combination of driving direction, steering angle, and orientation of the wheelchair relative to the map, the vehicle's movement was precalculated. Using the robot simulator SimRobot [17, 15], the cells of the map that would be visited by the wheelchair were determined. The travel distance to be checked for each direction/angle/orientation-combination is prescribed by the parameters of the wheelchair's braking distance, its response delay to commands, and its minimum safety distance to surrounding objects. In real-time systems, it is a common approach to waste the resource of memory for the sake of saving operation time. As a result of the simulated movements, we know in each situation which cells in the local obstacle map have to be checked for objects and how far the wheelchair has to drive to reach a cell. Since this is a means to determine the distance to obstacles that works on an internal representation of the environment, i. e. the map, it is called a *virtual* sensor.

The shape of a virtual sensor is not only determined by the three variables driving direction, steering angle and orientation of the wheelchair but also by various constants that have to be considered during the simulation: the overall stopping distance, the potential change of the steering angle in time, and quantization problems when mapping the shape of the virtual sensor into the grid of the map. Thus, each virtual sensor anticipates the wheelchair's worst-case movement for a range of rotations and steering angles to retain safety despite the loss of precision due to the use of a discrete map.

Optimizing Processing Time

To speed up searching the map, the internal data structure of a virtual sensor consists of a list of the indices (relative addresses) of the relevant cells. In addition, these indices are sorted by the distance the wheelchair has to move to reach the corresponding cell. As a result, the function that returns the distance to the closest object on the desired trajectory (specified by intended driving direction, intended steering angle and current orientation of the wheelchair) simply has to step through the list of cell references of the virtual sensor. As the list is ordered, the first entry found that refers to an occupied cell in the map always represents the closest object to the wheelchair. Since the distance is also stored in the virtual sensor, it need not to be computed at runtime but is immediately known. Searching a virtual sensor with a length of 150 cm takes less than 100 μs, i. e. more than 100 times faster than measuring such a distance with a real sonar sensor.

3.3 Adaptive Sonar Fire Strategy

As has been described in section 2, the use of a static strategy for firing the ultrasonic sensors involves the risk of overlooking relevant obstacles. Therefore, such a strategy is unsuitable to acquire readings for a map that is employed for collision avoidance. In contrast, a dynamic strategy is used, that determines the fire sequence from the wheelchair's current movement direction, its steering angle, and the data in the map. As stated in section 3.2, the calculation of the distance to the closest obstacle expects that all relevant data is already represented in the map when the collision distance should be determined. Therefore, two mechanisms were implemented to measure the relevant parts of the environment as fast as possible, and to enter their readings into the map:

Relevant Area

The wheelchair uses only the sensors that "see" the areas of the environment, which are relevant for the collision avoidance in the current situation, i. e. the areas covered by the actual virtual sensor (cf. Fig. 3). If there are obstacles in the area searched by the virtual sensor, only those cells are interesting that are closer than the nearest known obstacle, because the vehicle already has to stop in front of the closest obstacle, and therefore farther obstacle are not relevant for stopping in time. In situations that are rich of obstacles, this further reduces the number of cells to be measured, and thus the number of sonars to be fired. So, the measurement update rate increases when the environment gets narrow. This is a desirable feature for a reactive system.

Measurement Age

To get a uniform coverage of measurements in the relevant area, the age of the last measurement is stored for each cell in the map (cf. Fig. 3). Thereby, the sensors can be fired that are orientated towards the cells that have not been measured for the longest period of time. As obstacles are only noticed by the collision avoidance when they were measured at least twice, their second detection is forced by setting the age of all cells to "very old" that were measured as occupied for the first time. Thus, these areas are immediately re-measured, either confirming their occupancy or rejecting it. Cells that are "scrolled in" from the borders of the map due to the wheelchairs movement, and thus have never been measured before, are assumed to be "old". During each measurement cycle, all cells age by one until they are re-measured or they reached the "old" state. Note that the "very old" state is reserved for forcing certain measurements, i. e. no cell can reach this state by aging.

3.4 Map Entries

For each cell in the map, the age of the last measurement is stored. It is represented as a number ranging from 0 ("new") to 62 ("old"), and the special state "very old", i. e. represented by 63. In addition, the occupancy state of the cell is represented. There are four possible states (cf. Fig. 3):

Never measured. The cell has not been measured yet, e. g., because it was scrolled into the map from the border. As the place represented by such a cell may contain an obstacle, the cell is treated by the collision avoidance as if it would be occupied, i. e. the wheelchair will stop in front of it. However, as such a cell is always "old", it will be measured quickly, and thus it will lose the "unmeasured" state before it will be treated as an obstacle.

Figure 4: The Bremen Autonomous Wheelchair "Rolland" at the Hannover Trade Fair 1999.

Empty. There is no obstacle in the area represented by the cell.

Obstacle supposed. The cell was measured as occupied once. In this case, the measurement's age is always "very old".

Obstacle confirmed. The cell is occupied, i. e. an obstacle was measured more than once at the corresponding real world position. Therefore, it is treated as a thread for the wheelchair.

3.5 Entering Readings into the Map

When a new sonar measurement has been taken, this reading is inserted into the map. Inside the opening angle of the sensor, all cells are cleared that are closer to the sensor than the distance measured but not closer than the minimum sensor range of 15 cm. The cells at the measured distance are marked as occupied. All other cells remain unchanged. In addition, the ages of all modified cells are changed. The age of the cells that were measured as occupied for the first time is set to "very old", the ages of all other cells are reset to "new."

To speed up the operation, the indices of the map cells are pre-calculated for each of the 27 ultrasound sensors and all possible rotations of wheelchair in the map. The indices are sorted by the cells' distances to the sensors. Thus, the readings can easily be entered into the map.

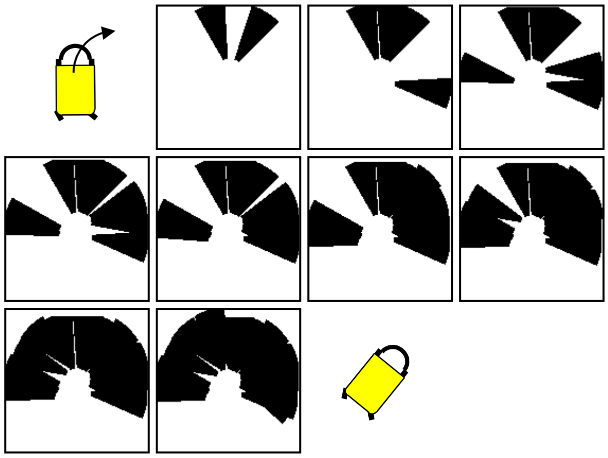

Figure 5: The adaptive sonar fire strategy. The relevant parts of the environment are measured faster than in Fig. 2.

3.6 Results

Figure 5 shows a result of the adaptive fire strategy. As in Fig. 2, the wheelchair performs a narrow curve to the right, but the fire strategy differs heavily: on the one hand, only those sensors are used that look towards the front of the vehicle and towards the outside of the curve, i. e. the areas the wheelchair may visit during driving the curve. On the other hand, this relevant environment is measured very quickly. Already after six measurement cycles, the spatial coverage is nearly perfect. The hole on the left side is not important at this time, because the wheelchair will not reach this area with its front. Only when the vehicle has turned further, it will contact the region with its back, and therefore it is measured then.
The quick update of sonar readings allows the user of the wheelchair to even drive in crowds of people. Although the calculation of the distance to the closest obstacle does not take dynamic objects into account, the wheelchair is able to cope with "cooperative" movements in the surroundings, i. e. obstacles that move but that do not actively collide with the wheelchair. This was presented at the Hannover Trade Fair 1999 (cf. Fig. 4).
The whole collision avoidance system—entering new sensor readings into the map, shifting the map, aging the measurements—is done in less than 3 ms on a Pentium 233 PC. As the wheelchair accepts drive commands every 32 ms, the map is updated 31 times per second.

4. OBSTACLE AVOIDANCE

Obstacle avoidance is one of the most important skills an autonomous mobile system should be able to rely on. Consequently, many different methods have been proposed in the literature. A brief survey is given in [3], where the authors distinguish between three classes of algorithms: edge-detection methods, approaches using certainty grids for obstacle representation and potential field methods. All of them are applicable with sonar sensors, but none of them is designed for use in a shared-control system.
The following subsections show why these systems are somewhat special, what design criteria should be met and how these requirements were coped with in the Bremen Autonomous Wheelchair.

4.1 Shared-Control

If a human operator is introduced into the system, the task of obstacle avoidance in mobile robots becomes even more challenging. The robot is no longer controlled by a software module (e. g. a path planner) that generates the motor command, but it is jointly controlled by the operator *and* the software module. In such shared-control environments it is often the case that a technical system monitors the commands issued by a human operator and intervenes if an intended action may cause some danger to the vehicle. As an example, consider the anti-lock braking system (ABS) which is used in cars to avoid the blocking of wheels in case of an emergency brake. If one wheel stops turning, the ABS releases the brake for a short time. Thus, the car remains maneuverable.
Depending on the amount of control left to the operator, various supervisory modes of the robot can be distinguished [16]: from a *manual supervisory mode* (car driver in control) to an *automatic supervisory mode* (ABS in control in case of blocked wheels).
The Bremen Autonomous Wheelchair comprises a hierarchy of four different supervisory modes:

Operator Control. The wheelchair is entirely controlled by the human operator.

Speed Controller. The assistive system reduces the speed according to the distance between an obstacle and the vehicle. The operator remains in control with respect to the steering as well as with respect to the maximum speed (the system will never issue a speed command faster than the one desired by the operator).

Obstacle Avoidance. The assistive system is responsible for the complete control of the wheelchair. It determines the target speed and the steering angle. The operator controls the maximum speed and sets a rough travel direction.

Automatic Behavior. When carrying out an automatic behavior, such as wall following or turning on the spot, the assistive system is in full control. The operator's only way of intervention is to stop the execution of these behaviors.

The following scenario describes a common situation in the daily life of a potential user of our system:

The person steers the wheelchair from the window through the living room and heads to a doorway. In front of the doorway the assistive system decelerates, and additionally, takes control of the steering. The vehicle travels safely through the doorway. The operator chooses the wall following behavior in order to drive through the corridor of his or her apartment.

Note that this scenario contains all the supervisory modes mentioned above. Traveling through the living room is done by the operator with no support by the assistive system (*Operator Control*). Close to the doorway the *Speed Controller* decelerates the vehicle. Obviously, an implicit mode transition took place. The same happens when the *Obstacle Avoidance* mode guides the wheelchair through the doorway. The operator cannot force this mode transition, it is controlled by the assistive system. In contrast, the switching to the *Automatic Behavior* mode has to be explicitly issued by the operator.

The distinction between different kinds of mode transitions is very important because it nourishes the hypothesis that designing an obstacle avoidance approach for a robot that is jointly controlled by a human and a machine is inherently different from doing so for an autonomous robot.

As an example, consider a high-level path planner of an autonomous robot which will always know the state of the underlying obstacle avoidance module, provided that this module indicated the relevant mode transitions. Thus, the path planner can adapt its behavior correspondingly. In contrast, a human operator may miss one or the other mode transition. Instead of waiting for the completion of an obstacle avoidance maneuver he or she may issue commands that contradict those desired by the assistive system. If the operator is not aware of the state of the technical system, a *mode confusion problem*, as discussed in [12], has occurred.

4.2 Design Criteria for Obstacle Avoidance

There are three principal decisions to be taken by an obstacle avoidance algorithm:

When should it change the steering angle?

Does the operator cope with detouring the obstacle, or does he or she need help. A design decision to be taken here is, whether the operator has to choose explicitly a distinct obstacle avoidance mode according to the planned maneuver. As an example consider the doorway passage which is nothing else than to detour the door posts. If the obstacle avoidance module intervenes too early, the wheelchair will probably fail to pass the doorway because its angle to the opening will be too bad.

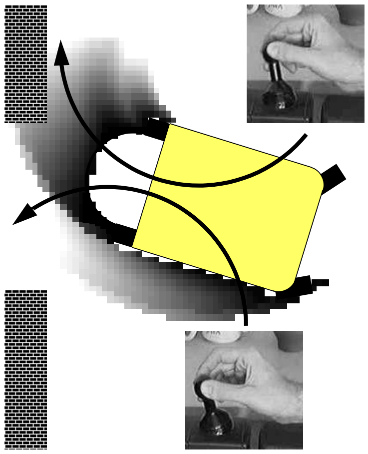

Figure 6: Deciding on which side the user wants the obstacle to be passed.

How should it alter the steering angle?

If heading towards an obstacle, the algorithm has to detour the obstacle either on the left or on the right side. Besides, it may depend on the situation if the steering angle should be changed as much as physically possible or only less. In a human-driven robot like the Bremen Autonomous Wheelchair, this is a question of comfort of traveling.

When should it additionally reduce speed?

The less the wheelchair is heading perpendicular towards an obstacle, the more it is sufficient to merely change the steering angle. As only a slight curve is required to detour the obstacle there is no need for a reduction of speed. On the other hand, there are situations in which it is indispensable to significantly decelerate in order to improve the ratio of steering speed to traveling speed.
The following subsection describes the shared-control obstacle avoidance approach used for the Bremen Autonomous Wheelchair. It takes the considerations presented here into account insofar as it gives the human operator as much control as possible.

4.3 Approach

The basic idea of our obstacle avoidance method is to detour obstacles in a way that is most likely to be acceptable for the operator. By taking into account the desired traveling direction in terms of the radius indicated by the operator via the joystick, it is decided whether the wheelchair should steer to the right or to the left.
A typical situation is depicted in Fig. 6: The wheelchair is placed in front of a doorway. If it simply drove forwards, it would hit into the right door post. If the operator indicates a right turn (as shown in the upper right picture), the assistive system infers that he or she wants to pass the obstacle (i. e. the

right door post) on the right-hand side. Thus it reinforces its steering angle to the right. If, instead, the operator indicates a left curve (as shown in the lower right picture), the assistive system takes it for granted that the operator wants to pass the obstacle on the left-hand side, i. e. to pass through the doorway.

In a former approach we implemented a rather machine-oriented obstacle avoidance behavior (cf. [11]). Similar to the vector field histogram (VFH) method [1], the best steering angle was chosen by searching the local obstacle map presented above for obstacle-free regions. Even though some enhancements in comparison to Borenstein's VFH method were implemented, this approach suffered from the fact that the success of an obstacle avoidance maneuver heavily depended on the time of intervention. If the assistive system took control of the wheelchair at a distance of 150 cm to an obstacle, it perfectly traveled along corridors and comfortably detoured cupboards, but it would have never passed through doorways.

Taking the experiences gathered with this primitive approach into account, the following obstacle avoidance method has been developed:

All the time, the obstacle avoidance module checks whether there are obstacles in the surrounding of the wheelchair it will collide with if it continues its current movement. The answer to this question is computationally very cheap, because it can be directly read from the virtual sensor that corresponds to the current steering angle. Due to the structure of the virtual sensors, the obstacles close to the wheelchair are found very quickly. Note that "close to the wheelchair" does not refer to Euclidean distance but to travel distance before collision.

As mentioned above, in common obstacle avoidance approaches it is a question of parameter optimization and intuition when the robot decides to detour a detected obstacle. Our approach pursues another strategy: in every operation step the minimum speed v_{mc} that leads to a collision with the detected closest obstacle is determined. As a result, reducing the current speed below this minimum collision speed v_{mc} will avoid the collision.

v_{mc} is not calculated during program execution but it is read out of a pre-calculated map, as depicted in Fig. 7. This map represents a lookup table for the function

$$v_{mc}(pos_{obs}, \alpha, dir)$$

where pos_{obs} is the position of the obstacle relative to the wheelchair, α is the current steering angle and dir encodes whether the wheelchair tries to pass the obstacle on the left or on the right side. The lookup table was calculated in advance by the simulation SimRobot that was mentioned above.

To determine the obstacle avoidance maneuver that

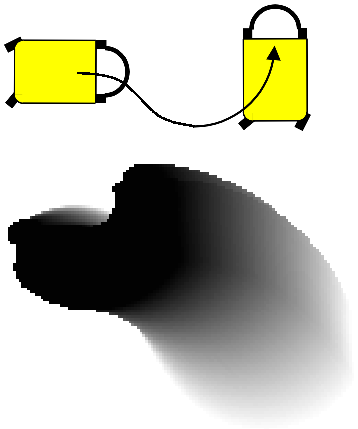

Figure 7: Avoid speed map. For each cell in the map, the map contains the smallest speed, at which the wheelchair will still collide with the cell while changing the steering angle from maximum right to maximum left until it turned by 90° to the left. The speed is depicted in grayscale; the brighter the cell, the faster the wheelchair can drive.

best fits the intention of the human operator, the radius indicated by the joystick is considered. If the detected obstacle is placed left of the described trajectory, it is assumed that the human operator wants to pass the object on the right side. If, instead, the indicated radius directly points towards the obstacle, the obstacle avoidance module does not intervene at all. This is because in such a case it is likely that the operator's intention was to move close to that obstacle and not to pass around it, e. g. when docking to a desk.

In addition, the direction of the detour (either left or right) also depends on the position of the closest obstacle relative to the wheelchair's front axle and its current driving direction (either forwards or backwards). As the wheelchair is driven by its front axle and steered by its rear axle, it detours an obstacle that is front left of it by a *right* turn. In contrast, an obstacle that is on the left side but behind the front axle has to be avoided by a *left* curve.

These criteria yield the detour directions that are depicted in Table 1 for forward movements, and in Table 2 for backward motion. For example, if the wheelchair drives forwards and the radius indicated by the operator passes the closest obstacle (which is in front of the front axle on the left-hand side of the vehicle) on the left side, the obstacle avoidance module decides to reinforce the left curve.

This method delivers contradictory commands if two different obstacles simultaneously collide with the body of the wheelchair, e. g. one in the front and one on the outer side. If these obstacles are either all in front of the driving axle or all behind it, they are treated like a single object that represents the average distance of these obstacles from the indicated radius. If there are some obstacles in front of the

Table 1: *Avoid direction when driving forwards.*

obstacle position	indicated direction left	none	right
front, left	left	straight	right
front, right	left	straight	right
rear, left	–	straight	left
rear, right	right	straight	–

Table 2: *Avoid direction when driving backwards.*

obstacle position	indicated direction left	none	right
front, left	–	straight	–
front, right	–	straight	–
rear, left	left	straight	right
rear, right	left	straight	right

driving axle and some behind it, the algorithm pays only attention to the former ones.
The obstacle avoidance approach described here deals with the three design criteria discussed in subsection 4.2 as follows:

- The obstacle avoidance module changes the steering angle at the latest possible point in time with respect to the actual speed. So, if at any time the wheelchair travels as fast as or faster than the smallest speed at which the vehicle would collide with the closest obstacle, the speed is reduced.

- The decision about the avoidance maneuver is made according to the rules shown in Tab. 1 and 2.

- The additional reduction of speed is also dependent on the detected obstacles.

4.4 Results

The obstacle avoidance approach was successfully tested in several settings including navigation in narrow corridors, doorway passage, and driving in environments crowded by people. For example, in Fig. 8 the wheelchair heads towards a doorway (depicted by two door posts, 1 m wide). Each of the outlines plotted in the figure corresponds to the wheelchair's position in 0.5 s steps. During the whole experiment, the user indicated with the joystick to drive straight ahead. As the desired travel direction is right of the closest obstacle (the left door post), the vehicle modifies the drive command to a the right curve. As a strong avoidance maneuver is required, the wheelchair additionally reduces the speed to achieve a better ratio of travel speed to steering speed. Thus it can pass the left door post, resulting in a situation, in which the right door post becomes the closest obstacle. As it is placed on the right side of the indicated travel direction, the wheelchair steers back and passes it on the left side.

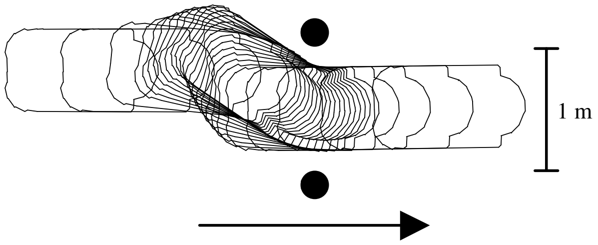

Figure 8: Experimental setup for an obstacle avoidance maneuver.

5. CONCLUSION AND FUTURE WORK

This paper proved that common static fire strategies for ultrasonic sensors are inherently unsafe as long as only a few sensors are fired simultaneously. A new adaptive strategy has been introduced that delivers a complete coverage of the robot's environment relevant for collision prevention. Furthermore, the general design problem for shared-control systems in human-oriented mobile robotics has been discussed. As a first result, a new obstacle avoidance approach for shared-control robotics systems has been presented.
Further research will concern an improvement of obstacle avoidance approach. Currently, the obstacle situation during avoidance maneuvers is always assessed from a local point of view. If the wheelchair detours an obstacle, the meaning of the travel direction indicated by the human operator changes, because the wheelchair's orientation shifts. So, in some cases, the vehicle passes a second obstacle on the wrong side, only because the user cannot adapt quickly enough to the situation that resulted from avoiding the first. To compensate for such shortcomings, the wheelchair should retain a global frame of reference as long as the operator does not intervene by changing the joystick position.

Acknowledgments

This work is supported by the Deutsche Forschungsgemeinschaft through the priority program "Spatial Cognition".

REFERENCES

[1] D. A. Bell, S. P. Levine, Y. Koren, L. A. Jaros, and J. Borenstein. Design criteria for obstacle avoidance in a shared control system. In *Pro-*

ceedings of the RESNA '94 Conference, pages 17–24, Nashville, 1994.

[2] J. Borenstein, H. R. Everett, and L. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning.* University of Michigan, 1996.

[3] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

[4] W. Burgard, D. Fox, and D. Henning. Fast grid-based position tracking for mobile robots. In G. Brewka, Ch. Habel, and B. Nebel, editors, *KI-97: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 289–300, Berlin, Heidelberg, New York, 1997. Springer.

[5] Th. Edlinger and G. Weiß. Exploration, navigation and self-localization in an autonomous mobile robot. In R. Dillmann, U. Rembold, and T. Lüth, editors, *Autonome Mobile Systeme*, Informatik aktuell, pages 142–151, Berlin, Heidelberg, New York, 1995. Springer.

[6] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots*, volume 1, pages 60–70, Los Alamitos, California, 1991. IEEE Computer Society Press.

[7] R. Furtwängler, U. D. Hanebeck, and G. Schmidt. Dynamic control of a mobile manipulator. In *Proceedings of AMC'98, 5th Int. Workshop on Advanced Motion Control*, pages 440–445, Coimbra, Portugal, 1998.

[8] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, Canada, 1998.

[9] A. Kirchner. Mapping with laser rangefinder and a grid based transformation estimation. In *Proc. Third ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems*, pages 320–325, Bremen, 1997.

[10] B. Krieg-Brückner, Th. Röfer, H.-O. Carmesin, and R. Müller. A taxonomy of spatial knowledge and its application to the Bremen Autonomous Wheelchair. In Ch. Freksa, Ch. Habel, and K. F. Wender, editors, *Spatial Cognition*, volume 1404 of *Lecture Notes in Artificial Intelligence*, pages 373–397, Berlin, Heidelberg, New York, 1998. Springer.

[11] A. Lankenau, O. Meyer, and B. Krieg-Brückner. Safety in robotics: The Bremen Autonomous Wheelchair. In *Proceedings of AMC'98, 5th Int. Workshop on Advanced Motion Control*, pages 524–529, Coimbra, Portugal, 1998.

[12] Nancy G. Leveson, L. Denise Pinnel, Sean David Sandys, Shuichi Koga, and Jon Damon Reese. Analyzing software specifications for mode confusion potential. In *Workshop on Human Error and System Development*, Glasgow, 1997.

[13] C. Owen and U. Nehmzow. Middle scale navigation – a case study. In *Spatial Reasoning in Mobile Robots and Animals*, pages 104–112, Manchester University, 1997. AISB-97 Workshop.

[14] Th. Röfer and A. Lankenau. Architecture and applications of the Bremen Autonomous Wheelchair. In P. P. Wang, editor, *Proc. of the Fourth Joint Conference on Information Systems*, volume 1, pages 365–368. Association for Intelligent Machinery, 1998.

[15] Th. Röfer. Strategies for using a simulation in the development of the Bremen Autonomous Wheelchair. In R. Zobel and D. Moeller, editors, *Simulation-Past, Present and Future*, pages 460–464. Society for Computer Simulation International, 1998.

[16] Nadine Sarter and David Woods. How in the world did we ever get into that mode? mode error and awareness in supervisory control. *Human Factors*, 37(1):5–19, 1995.

[17] U. Siems, Ch. Herwig, and Th. Röfer. Simrobot - Ein Programm zur Simulation sensorbestückter Agenten in einer dreidimensionalen Umwelt. In B. Krieg-Brückner, G. Roth, and H. Schwegler, editors, *ZKW-Bericht*, number 1/94 in ISSN 0947-0204, Universität Bremen, 1994. Zentrum für Kognitionswissenschaften. Also: http://www.tzi.org/~simrobot.