Controlling Synchro-drive Robots with the Dynamic Window Approach to Collision Avoidance

Dieter Fox[†], Wolfram Burgard[†], and Sebastian Thrun[†‡]
[†]Dept. of Computer Science III, University of Bonn, D-53117 Bonn
[‡]Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, P A 15213
Email: {fox,wolfram}@uran.cs.uni-bonn.de, thrun@cs.cmu.edu

Abstract

This paper proposes the dynamic window approach to reactive collision avoidance for mobile robots equipped with synchro-drives. The approach is derived directly from the motion dynamics of the robot and is therefore particularly well-suited for robots operating at high speed. It differs from previous approaches in that the search for commands controlling the translational and rotational velocity of the robot is carried out directly in the space of velocities. The advantage of our approach is that it correctly and in a rigorous way incorporates the dynamics of the robot. This is done by reducing the search space to the dynamic window*, which consists of the velocities reachable within a short time interval. Within the dynamic window the approach only considers admissible velocities yielding a trajectory on which the robot is able to stop safely. Among these velocities the combination of translational and rotational velocity is chosen by maximizing an objective function. The objective function includes a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory. In extensive experiments the approach presented here has been found to safely control our mobile robot RHINO with speeds of up to 95 cm/sec, in populated and dynamic environments.*

1 Introduction

One of the ultimate goals of indoor mobile robotics research is to build robots that can safely carry out missions in hazardous and populated environments. For example, a service-robot that assists humans in indoor office environments should be able to react rapidly to unforeseen changes, and perform its task under a wide variety of external circumstances. Most of today's commercial mobile devices scale poorly along this dimension. Their motion planning relies on accurate, static models of the environments, and therefore they often seize to function if humans or other unpredictable obstacles block their path. To build autonomous mobile robots one has to build systems that can perceive their environments, react to unforeseen circumstances, and (re)plan dynamically in order to achieve their missions.

This paper focuses on one particular aspect of the design of such a robot: the avoidance of collisions with obstacles. In general, we desire an approach to collision avoidance to have the following properties:

1. **Safe.** The robot must travel safely even with high speed. It therefore must take the dynamic constraints into account.
2. **Reactive.** The robot should react adequately and rapidly to unforeseen circumstances. This requires fast techniques for the detection of obstacles and the selection of appropriate steering commands.
3. **Effective.** The robot should make maximum progress towards the goal. This implies that whenever advantageous, the robot should modify its travel direction to stay away from obstacles.

Unlike most previous approaches, the *dynamic window approach* proposed here meets all these requirements. It is safe, because it rigorously takes the inertia and the torque limits into account. It quickly reacts to unforeseen obstacles, and re-plans dynamically, trading off direct progress towards the goal and clearance.

The dynamic window approach is based on an approximation of the exact motion equations of synchro-drive robots by sequences of sequences of circular arcs [4]. In a nutshell, the dynamic window approach works as follows. It first prunes the overall search space of trajectories, resulting in a two-dimensional space of circular trajectories. Then, the search space is reduced to the admissible velocities allowing the robot to stop safely without colliding with an obstacle. Finally, the *dynamic window* restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot. This way the dynamic constraints are properly taken into account. After that, the robot picks a trajectory at which it can maximize its translational velocity and the distance to obstacles, yet minimize the angle to its goal relative to its own heading direction. This is done by maximizing an appropriate objective function. The combination of all objectives leads to a very robust, efficient, and reactive collision avoidance strategy.

Figure 1. The robot RHINO, an RWI B21.

The dynamic window approach has been implemented and tested using RHINO, a B21 robot manufactured by Real World Interface Inc. (see Figure 1), and other synchro-drive robots. In extensive experimental evaluations using ultrasonic proximity sensors for the construction of local world models (obstacle line fields), the method has proven to avoid collisions reliably with speeds of up to 95 cm/sec on several robots in several indoor environments (University of Bonn, Carnegie Mellon University, 1994 AAAI robot competition, 1995 IJCAI robot exhibition, and others, see also [3]). The method has also successfully been operated based on cameras and infrared detectors as sensory input.

Our approach differs from previous approaches in (a) that it is derived directly from the motion dynamics of a mobile robot, (b) it therefore takes the inertia of the robot into account – which is particularly important if a robot with torque limits travels at high speed –, and (c) has safely controlled several RWI robots in various cluttered and dynamic environments with speeds of up to 95 centimeter per second. We envision this approach to be particularly useful for robots that travel at even higher speeds and for low-cost robots with limited motor torques, for which the constraints imposed by the motion dynamics are even more imperative.

The remainder of this paper is organized as follows. After discussing related work Section 3 describes our approach, as outlined above. Experimental results are summarized in Section 4, followed by a discussion of further research issues.

2 Related Work

The collision avoidance approaches for mobile robots can roughly be divided into two categories: *global* and *local*. The global techniques, such as road-map, cell decomposition and potential field methods (see [8] for an overview and further references), generally assume that a complete model of the robot's environment is available. The advantage of global approaches lies in the fact that a complete trajectory from the starting point to the target point can be computed off-line. However, global approaches are not appropriate for fast obstacle avoidance. Their strength is global path planning. More specifically, these methods have proven problematic when the global world model is inaccurate, or simply not available, as is typically the case in most populated indoor environments. Hu/Brady, Moravec and others [5, 9], have shown how to update global world models based on sensory input, using probabilistic representations. A second disadvantage of global methods is their slowness due to the inherent complexity of robot motion planning [10]. This is particularly problematic if the underlying world model changes on-the-fly, because of the resulting need for repeated adjustments of the global plan. In such cases, planning in a global model is usually too expensive to be done repeatedly.

Local approaches, on the other hand, are used for reactive collision avoidance. They consider only a small fraction of a temporary world model to generate robot control. This comes at the obvious disadvantage that they cannot produce optimal solutions. Local approaches are easily trapped in local minima (such as U-shaped obstacle configurations). However, the key advantage of local techniques over global ones lies in their low computational complexity, which is particularly important when the world model is updated frequently based on sensor information. For example, potential field methods, as proposed by [6], determine the steering direction by (hypothetically) assuming that obstacles assert negative forces on the robot, and that the target location asserts a positive force. These methods are extremely fast, and they typically consider only the small subset of obstacles close to the robot. Borenstein and Koren [7] identified that such methods often fail to find trajectories between closely spaced obstacles; they also can produce oscillatory behavior in narrow corridors. In [2], the *vector field histogram* approach is proposed, which extends the previously developed *virtual force field histogram* [1]. This approach uses an occupancy grid representation for modeling the robot's environment, which is generated and updated continuously using ultrasonic proximity sensors. Occupancy information is transformed into a histogram description of the free space around the robot, which is used to compute the motion direction and velocity for the robot. In [11] a method similar to ours has been proposed which, according to Simmons, has been developed later but independently. As noted above, local methods are typically very fast, and they quickly adapt to unforeseen changes in the environment.

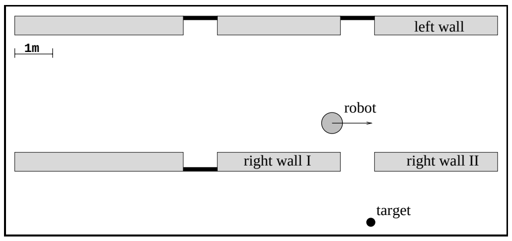

Figure 2. Example situation

Current approaches deal poorly with the constraints imposed by the dynamics of the robot. This is due to the fact that most of the local approaches generate motion commands for the robot in two separate stages [1, 2, 6]. In the first stage a desired motion direction is determined. In the second stage the steering commands yielding a motion into the desired direction are generated. Strictly speaking, such an approach is only legitimate if infinite forces can be asserted on the robot. However, for robots with limited accelerations it is necessary to take into account the impulse of the robot.

For example consider the situation given in Figure 2 and suppose that the robot is in a fast straight motion through the corridor while the target point is in the small opening to its right. Obviously, the optimal target direction implies a turn to the right. However, a robot whose forces are not high enough to perform the necessary sharp turn would collide with the wall (*right wall II*) if it does not consider its dynamics constraints. The *dynamic window approach* is especially designed to deal with such situations. By only considering the admissible velocities in the dynamic window the robot detects that it cannot turn to the right and therefore keeps its straight motion direction.

3 The Dynamic Window Approach

In [4] we showed how to approximate the trajectory of a synchro-drive robot by a sequence of circular arcs. In the remainder of this paper we will refer to these circles as curvatures. Each curvature is uniquely determined by a velocity vector (v, ω) of translational velocity v and rotational velocity ω. In the dynamic window approach the search for commands controlling the robot is carried out directly in the space of such velocities. The dynamics of the robot is incorporated into the method by reducing the search space to those velocities which are reachable under the dynamic constraints. In addition to this restriction only velocities are considered which are safe with respect to the obstacles. This pruning of the search space is done in the first step of the algorithm. In the second step the velocity maximizing the objective function is chosen from the remaining velocities. A brief outline of the different parts of one cycle of the algorithm is given in the following box (in the current implementation such a cycle is performed every 0.25 seconds).

1. **Search space:** The search space of the possible velocities is reduced in three steps:
   (a) **Circular trajectories:** The dynamic window approach considers only circular trajectories (curvatures) uniquely determined by pairs (v, ω) of translational and rotational velocities. This results in a two-dimensional velocity search space.
   (b) **Admissible velocities:** A pair (v, ω) is considered admissible, if the robot is able to stop before it reaches the closest obstacle on the corresponding curvature. The restriction to admissible velocities ensures that only safe trajectories are considered.
   (c) **Dynamic window:** The dynamic window restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot.
2. **Optimization:** The objective function

   $$\begin{aligned} G(v,\omega) &= \sigma(\alpha \cdot \text{angle}(v,\omega) + \beta \cdot \text{dist}(v,\omega) \\ &\qquad + \gamma \cdot \text{vel}(v,\omega)) \end{aligned}$$

   is maximized. With respect to the current position and orientation of the robot this function trades off the following aspects:
   (a) **Target heading:** *angle* is a measure of progress towards the goal location. It is maximal if the robot moves directly towards the target.
   (b) **Clearance:** *dist* is the distance to the closest obstacle on the trajectory. The smaller the distance to an obstacle the higher is the robot's desire to move around it.
   (c) **Velocity:** *vel* is the forward velocity of the robot and supports fast movements.

   The function σ smoothes the weighted sum of the three components and results in more side-clearance from obstacles.

A more detailed description is given in Sections 3.1-3.2 In the remainder of this section we will use the situation shown in Figure 2 to describe the different aspects of the dynamic window approach.

3.1 Search Space

(a) Circular trajectories

To make the search for velocity commands feasible, the dynamic window approach considers exclusively the *next* velocity, and assumes that the velocities in the remaining time intervals are constant (which is equivalent to assuming zero accelerations after the first time intervals). This reduction is motivated by the observations that (a) the reduced search space is two-dimensional and thus tractable, (b) the search is repeated after each time interval, and (c) the velocities will automatically stay constant if no new commands are given.

(b) Admissible Velocities

Obstacles in the closer environment of the robot impose restrictions on the rotational and translational velocities. For example, the maximal admissible speed on a curvature depends on the distance to the next obstacle on this curvature. Assume that for a velocity (v, ω) the term *dist*(v, ω) represents the distance to the closest obstacle on the corresponding curvature (in Section 4.1 we describe how to compute this distance given circular trajectories). A velocity is considered admissible, if the robot is able to stop without collision. Let $\dot{v}_b$ and $\dot{\omega}_b$ be the accelerations for breakage. Then the set V_a of admissible velocities is defined as

$$\begin{aligned} V_a &= \Big\{(v,\omega) \mid v \leq \sqrt{2 \cdot \text{dist}(v,\omega) \cdot \dot{v}_b} \\ &\qquad \wedge\, \omega \leq \sqrt{2 \cdot \text{dist}(v,\omega) \cdot \dot{\omega}_b}\Big\}. \end{aligned}$$

Thus V_a is the set of velocities (v, ω) allowing the robot to stop without colliding with an obstacle. Consider the example given in Figure 2. Figure 3 shows the velocities admissible in this situation given the accelerations $\dot{v}_b = 50$ cm/sec^2 and $\dot{\omega}_b = 60$ deg/sec^2. The non-admissible velocities are denoted by the dark shaded areas. For example, all velocities in area *right wall II* would cause a sharp turn to the right and thus cause the robot to collide with the right wall in the example situation. The non-admissible areas are obtained using sonar sensors (see Section 4).

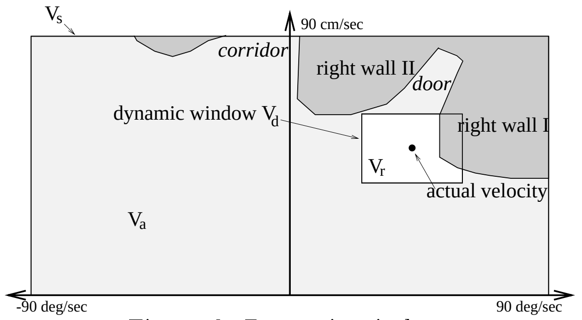

Figure 3. Dynamic window

(c) Dynamic window

To take into account the limited accelerations, the overall search space is reduced to those velocities that are attainable within the next time interval. Let t be the time interval during which the accelerations $\dot{v}$ and $\dot{\omega}$ will be applied and let (v_a, ω_a) be the actual velocity. Then the *dynamic window* V_d is defined as

$$\begin{aligned} V_d \;&=\; \big\{(v,\omega) \mid v \epsilon [v_a - \dot{v}\cdot t, v_a + \dot{v}\cdot t] \\ &\qquad \wedge\, \omega \epsilon [\omega_a - \dot{\omega}\cdot t, \omega_a + \dot{\omega}\cdot t]\big\}. \end{aligned}$$

The dynamic window is centered around the actual velocity and the extensions of it depend on the accelerations that can be exerted. All curvatures outside the dynamic window cannot be reached within the next time interval and thus are not considered for the obstacle avoidance. An example dynamic window obtained in the situation shown in Figure 2 given a time interval of 0.25 sec is shown in Figure 3.

Resulting Search Space

The above restrictions of the search space lead to a smaller region in the velocity space, called V_r (see Figure 3). Let V_s be the space of possible velocities, then the area V_r is defined as the intersection of the restricted areas, namely $V_r = V_s \cap V_a \cap V_d$. In Figure 3 the resulting search space is represented by the white area.

3.2 Maximizing the Objective Function

Having determined the resulting search space V_r, a velocity is selected from V_r. To incorporate the criteria *target heading*, *clearance*, and *velocity*, the maximum of the objective function

$$\begin{aligned} G(v,\omega) \;&=\; \sigma(\alpha\cdot \mathrm{angle}(v,\omega) + \beta\cdot \mathrm{dist}(v,\omega) \\ &\qquad +\gamma\cdot \mathrm{velocity}(v,\omega)) \end{aligned}$$

is computed within V_r.

(a) Target heading

The target heading $angle(v,\omega)$ measures the alignment of the robot with the target direction. It is given by the angle of the target point relative to the robot's heading direction. Since this direction changes with the different velocities target heading θ is computed for a predicted position of the robot as shown in Figure 4.

The predicted position is the position reached by the robot if it moves with the selected velocity for the next time interval and then stops with maximal decelerations.

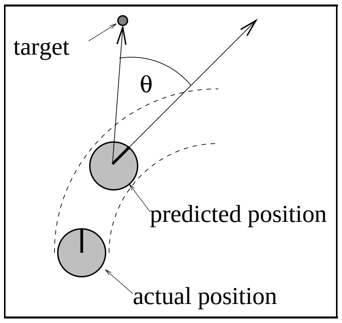

Figure 4. Target heading θ.

(b) Clearance

The function $dist(v,\omega)$ represents the distance to the closest obstacle that intersects with the curvature. If no obstacle is on the curvature this value is set to a large constant.

(c) Velocity

The function $velocity(v,\omega)$ is used to evaluate the progress of the robot on the corresponding trajectory. It is simply a projection on the translational velocity v.

Smoothing

All three components of the objective function are normalized to $[0,1]$ and locally smoothed. The smoothing increases side-clearance of the robot. The resulting objective function for our example is shown in Figure 5. In the plot the values for non-admissible velocities are set to zero (compare with Figures 2 and 3). For simplicity we show the evaluation of the entire velocity space and do not restrict it to a dynamic window. The function is obtained by a value of 0.2 for α and γ and a value of 2.0 for β. Due to the impact of the target heading velocities yielding a turn to the right get higher values. The position of the maximal value is depicted by the vertical line. As expected, the fastest trajectory leading through the door area has the largest value.

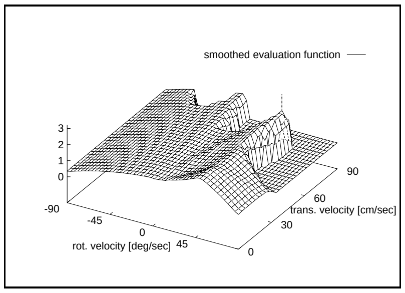

Figure 5. Objective function

It should be noticed that all three components of G, the target heading, the clearance, and the velocity, are necessary. By maximizing solely the clearance and the velocity, the robot would always travel into free space but there would be no incentive to move towards a goal location. By solely maximizing the target heading the robot quickly would get stopped by the first obstacle that blocks its way, unable to move around it. By combining all three components, the robot moves round obstacles as fast as it can under the constraints listed above, while still making progress towards the goal.

In our implementation, the objective function is maximized by discrete search in a 10×10 grid. This search is repeated every 0.25 seconds, which is frequently enough to yield smooth trajectories.

4 Implementation and Experimental Results

The dynamic window approach has been implemented and tested using the robot *RHINO* which is a synchro-drive robot currently equipped with a ring of 24 Polaroid ultrasonic sensors, 56 infrared detectors, and a stereo camera system. Because the main beam width of an ultrasonic transducer is approximately 15°, the whole 360° area surrounding the robot can be measured with one sweep of all sensors. A complete sonar sweep takes approximately 0.4 sec. The infrared detectors and the camera system are not used in the experiments reported here.

4.1 The Obstacle Line Field

The world is modeled by an obstacle line field [3], which is a two-dimensional description of sensory data relative to the robot's position (see Figure 6). We adjusted our sonar sensors such that most erroneous readings indicate a too long distance. To be maximally conservative, every reading is converted to an obstacle line. If the sensors would produce spurious short readings (e.g. due to cross-talk), more sophisticated sensor interpretation and integration models such as for example occupancy probability grid maps [9] would be required.

The obstacle line field is centered around the robot's position and is built out of the data gathered by proximity sensors. It contains a line for each reading of a sonar sensor, which is perpendicular to the main axis of the sensor beam at the measured distance. The length of the line is determined by the breadth of the beam in the given distance. Using this obstacle representation the distance to obstacles on curvatures is computed as follows: let r be the radius of the circular trajectory, and let γ be the angle between the intersection with

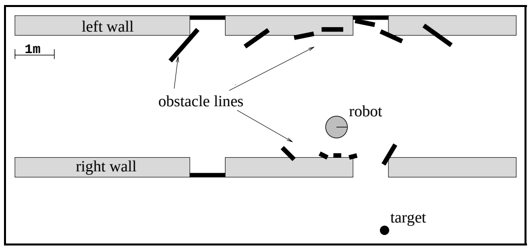

Figure 6. Corridor with exemplary obstacle lines and target point

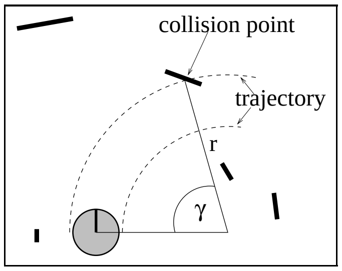

Figure 7. Determination of the distance

the obstacle line and the position of the robot (see Figure 7). Then the distance to the next obstacle is given by $\gamma \cdot r$.

To allow the robot to react quickly to changes in the environment, we limit the number of lines to 72 and apply a first-in-first-out strategy to remove the least actual lines from the obstacle line field. We found these values to allow the robot to travel safely even with speeds of up to 95 cm/sec, while simultaneously keeping the computational time within 0.25 sec on an i486 computer.

4.2 Further Implementation Details

The following additional strategies improved the maneuverability and the elegance of robot motion. Since they are not essential for the approach proposed in this paper, we will only sketch them here.

- **Rotate away mode.** In rare cases we observed that the robot got stuck in local minima. This is the case if no admissible trajectory allows the robot to translate. When this condition occurs, which is easily detected, the robot rotates until it is able to translate again.

- **Speed dependent side clearance.** To adapt the speed of the robot according to the side clearance to obstacles we introduced a safety margin around the robot, which grows linearly with the robot's translational velocity. Thus the robot travels with high speed through corridors and decelerates when driving through narrow doors.

4.3 Experimental Results

Based on the dynamic window approach to collision avoidance, *RHINO* has been operated safely in various environments, over the last 2 years. The method showed great reliability within extensive tests during more than 100 hours. Collisions were only caused by fast moving obstacles (e.g. doors) or sensor failures. The maximum velocity is constrained by the robot's hardware to approximately 95 cm/sec. *RHINO* reaches this velocity in large openings and hallways, when no obstacles block its way. If obstacles block its way, slower velocities are selected, and collisions are avoided by selecting appropriate trajectories. For example, when moving through doors, *RHINO* typically decelerates to approximately 20 cm per second in the vicinity of the door.

In the remainder of this section, we will give experimental results generated with the dynamic window approach. Although its performance depends on the weighting parameters α, β, and γ, it is robust with respect to slight changes of their values. Without any exhaustive tuning of these parameters we found values of 0.2, 0.2 and, 2.0 for α, β, and γ to give good results. In the following figures the environment is drawn by hand. Nonetheless each plot stems from an actual experiment and all shown trajectories are extracted from real position data. Each of these examples show the complete path to a particular goal point, which in our tests is set by a human operator. In the every-day use, these goal points are set automatically by a global path planner described in [12].

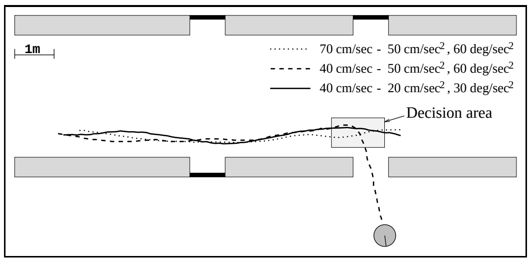
1m
70 cm/sec - 50 cm/sec², 60 deg/sec²
40 cm/sec - 50 cm/sec², 60 deg/sec²
40 cm/sec - 20 cm/sec², 30 deg/sec²
Decision area

Figure 8. Trajectories chosen for different dynamic parameters

Role of the Dynamic Window

The first experiment demonstrates the impact of the dynamic window on the behavior of the robot. The three paths in Figure 8 are examples for the typical behavior of the robot under different dynamics constraints. The crucial point of the experiment is the path taken in the dark shaded *decision area*. In this area the robot detects the opening to its right and has to decide whether to take the sharp turn to the right or not. This decision strongly depends on the dynamics of the robot. Only if the actual velocity and the possible accelerations allow the robot to turn sharply to the right, the robot directly moves to the target point. This trajectory is denoted by the dashed line. In the other two cases the robot decides to pass the opening and moves parallel to the wall until the evaluation of the target heading angle(v, ω) becomes very small. Notice that without considering the dynamic constraints, an attempt to turn right would have resulted in a collision with a wall. Other approaches, that do not consider dynamics, would be subject to such failures. In fact, in initial experiments with a simulator, in which we ignored some of the dynamic effects, we experienced such collisions frequently.

Straight Motion in Corridors

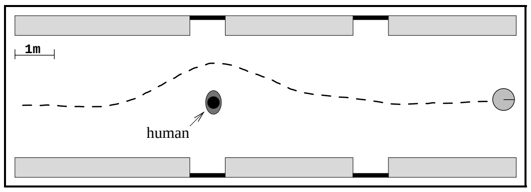

Figure 9. Trajectory through corridor

Figure 9 shows an example of traveling along a hallway with only one obstacle in the middle of the hallway. In this case *RHINO* first orients itself to the target point. But then the obstacle is detected and the robot chooses a smooth trajectory avoiding the obstacle. Although *RHINO* slows down to 55 cm/sec before passing the obstacle, the average speed in this experiment was approximately 72 cm/sec. It should be noted that after having driven round the obstacle *RHINO* follows straight lines whenever possible, and does not oscillate, as sometimes is the case with potential field approaches [7].

Fast Motion through Cluttered Environments

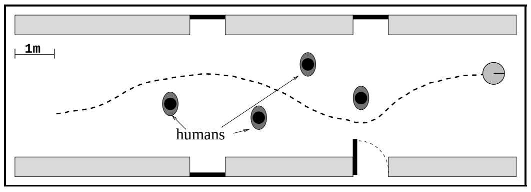

Figure 10. Trajectory through cluttered corridor

The third experiment is shown in Figure 10 and involves traveling through a cluttered corridor. All humans in the corridor are smoothly circumvented with a

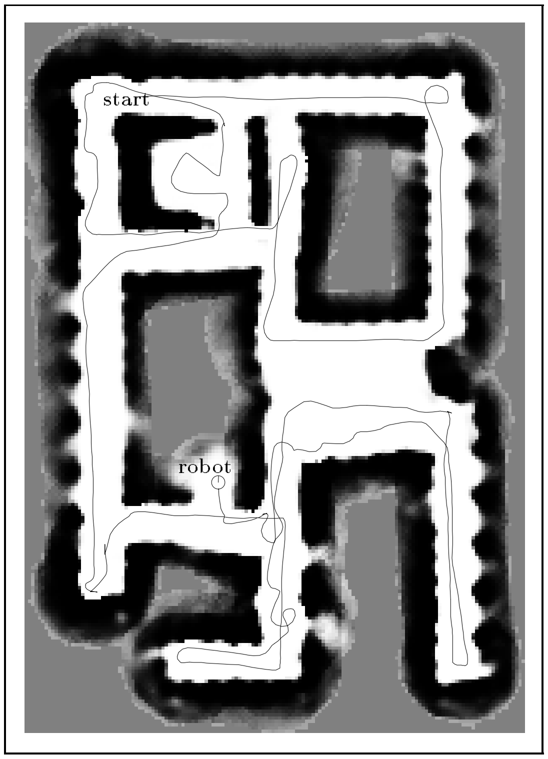

Figure 11. Run at the AAAI '94 mobile robot competition

maximal speed of 95 cm/sec. Notice that although the robot decelerates to less than 20 cm/sec when passing the narrow passage (less than 80 cm wide) between the fourth person and the open door, it still maintains an average speed of 65 cm/sec!

The AAAI '94 Mobile Robot Competition

The trajectory shown in Figure 11 was generated in the arena of the AAAI '94 mobile robot competition. The figure gives a plot of the occupancy map of the arena along with the trajectory of the robot. Here the robot moved free of collisions in an artificial indoor environment during an exploration run which took about 15 minutes. The target points for the collision avoidance were generated by a global planning algorithm. Doors were approximately 80-110 cm wide.

It is generally difficult to compare the results described here to results obtained by other researchers, mainly because robots vary in sizes, and small changes in the environment can have an enormous impact on the difficulty of the problem. For example, in a configuration similar to the ones shown in Figures 9 and 10 Borenstein et al. report that their robot traveled with an average speed of 58 cm/sec through a cluttered environment [1, 2]. As far as it can be judged from a single example (which is all that is available in [1, 2]), our results compare favorably to those of Borenstein et al.

5 Discussion

This paper describes the *dynamic window* approach to collision avoidance for mobile synchro-drive robots. As demonstrated by our extensive empirical tests, this approach is safe even at high speed, it reacts adequately, and it effectively steers the robot to its goal.

The key to the effectiveness of the approach lies in the objective function for the selection of motion commands. This function trades off a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory. By combining these, the robot trades off its desire to move fast towards the goal and its desire to ship around obstacles (which decrease the free space). Since dynamic constraints are taken into account via the dynamic window, the robot operates safely even at high speed. The experiments show that the combination of all objectives leads to a robust collision avoidance strategy that safely operates our robot *RHINO* with speeds of up to 95 cm/sec. Notice that the approach described here is only part of the overall *RHINO* control architecture. For example, approaches for building occupancy maps, global path planning and computer vision are surveyed in [3].

The technique of approximating the trajectories of mobile robots by circular arcs is not restricted to synchro-drive robots. If the influence of the shape of the robot on the curvatures depending on steering commands is taken into account, the dynamic window approach can also be applied to non-synchro-drive robots.

In principle, the approach proposed here only assumes geometric information about the relative location of obstacles. Therefore, it is well-suited for proximity sensors such as ultrasonic transducers, which were used in the experiments reported here, or laser range-finders. In some preliminary tests we also used camera and infrared sensors for the detection of obstacles. Knowing the geometry of the robot and the angle of its camera, pixel information was converted to proximity information. However, the resulting proximity estimate was only accurate if an obstacle extends to the floor. Obstacles in different heights lead to an overestimation of distance, which would cause the robot to collide. Stereo vision might potentially overcome this problem. The result with infrared detectors suffered from the fact that *RHINO*'s detectors have only a small range of view (< 30 cm). Therefore, when moving high-speed the robot may collide nonetheless. Combining either of the two sensor systems with ultrasonic measurements, however, consistently improved the smoothness of the robot's trajectories.

References

[1] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume CH-2876, pages 572–577, 1990.

[2] Johann Borenstein and Yoram Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[3] Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox, Thomas Hofmann, Frank Schneider, Jiannis Strikos, and Sebastian Thrun. The mobile robot Rhino. *AI Magazine*, 16(1), 1995.

[4] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13, University of Bonn, 1995. `http://www.cs.uni-bonn.de/~fox/`.

[5] Huosheng Hu and Michael Brady. A Bayesian approach to real-time obstacle avoidance for a mobile robot. In *Autonomous Robots*, volume 1, pages 69–92. Kluwer Academic Publishers, Boston, 1994.

[6] Oussama Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[7] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, April 1991.

[8] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991. ISBN 0-7923-9206-X.

[9] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[10] Jacob T. Schwartz, Micha Scharir, and John Hopcroft. *Planning, Geometry and Complexity of Robot Motion.* Ablex Publishing Corporation, Norwood, NJ, 1987.

[11] Reid Simmons. The curvature-velocity method for local obstacle avoidance. To appear in *Proceedings of ICRA '96.*

[12] Sebastian Thrun. Exploration and model building in mobile robot domains. In *Proceedings of the ICNN-93*, pages 175–180, San Francisco, CA, March 1993. IEEE Neural Network Council.