# The Dynamic Window Approach to Collision Avoidance


**Dieter Fox**[†] **Wolfram Burgard**[†] **Sebastian Thrun**[†‡]

[†]Dept. of Computer Science III, University of Bonn, D-53117 Bonn, Germany

[‡]Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

Email: {fox,wolfram}@uran.cs.uni-bonn.de, thrun@cs.cmu.edu



**Abstract**

This paper describes 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 an elegant 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 reactive avoidance of collisions with obstacles. The *dynamic window approach* proposed in this paper is especially designed to deal with the constraints imposed by limited velocities and accelerations, because it is derived directly from the motion dynamics of synchro-drive mobile robots. In a nutshell, our approach considers periodically only a short time interval when computing the next steering command to avoid the enormous complexity of the general motion planning problem. The approximation of trajectories during such a time interval by circular curvatures results in a two-dimensional search space of translational and rotational velocities. This search space is reduced to the admissible velocities allowing the robot to stop safely. Due to the limited accelerations of the motors a further restriction is imposed on the velocities: the robot only considers velocities that can be reached within the next time interval. These velocities form the *dynamic window* which is centred around the current velocities of the robot in the velocity space.

Among the admissible velocities within the dynamic window 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. 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). The combination of all objectives leads to a very robust and elegant 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 gives the general motion equations for synchro-drive mobile robots. One of the key results here is that trajectories of synchro-drive robots can be approximated accurately by finitely many segments of circles. Section 4 describes our approach, as outlined above. Experimental results are summarized in Section 5, 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 [10] 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. Additionally, 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, 11], 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 [12]. 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 or reactive approaches, on the other hand, use only a small fraction of the 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 [8], 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 [9] identified that such methods often fail to find trajectories between closely spaced obstacles; they also can produce oscillatory behavior in narrow corridors. An extended version of the potential field approach is introduced in [7]. By modifying the potential function the motion of the robot becomes more efficient and different behaviors such as wall following and tracking can be achieved.

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. As noted above, local methods are typically very fast, and they quickly adapt to unforeseen changes in the environment.

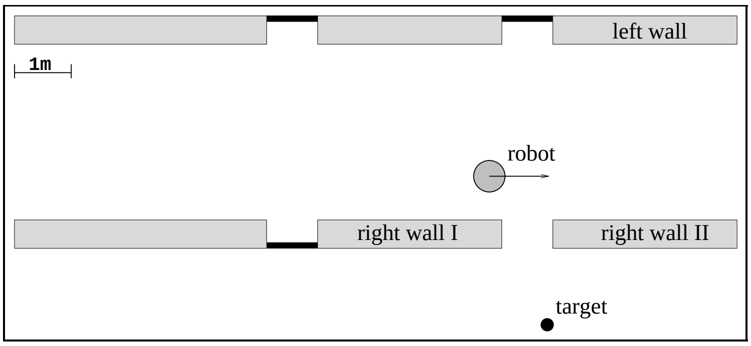


Figure 2. Example situation

Most of these local approaches generate motion commands for the robot in two separate stages [1, 2, 8]. 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 justifiable 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 in 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. Without respecting that its forces are not high enough to perform the necessary sharp turn the robot would collide with the wall (*right wall II*). By only considering the admissible velocities in the dynamic window our method detects that the robot cannot perform the sharp turn. Thus the robot would stay on its current straight trajectory and not collide with the wall.

# 3 Motion Equations for a Synchro-Drive Robot

This section describes the fundamental motion equations for a synchro-drive mobile robot [4]. The derivation begins with the correct dynamic laws, assuming that the robot's translational and rotational velocity can be controlled independently (with limited torques). To make the equations more practical, we derive an approximation that models velocity as a piecewise constant function in time. Under this assumption, robot trajectories consist of sequences of finitely many segments of circles. Such representations are very convenient for collision checking, since intersections of obstacles with circles are easy to check. We also derive an upper bound for the approximation error. The piecewise circular representation forms the basis of the dynamic window approach to collision avoidance, described in Section 4.

## 3.1 General Motion Equations

Let $x(t)$ and $y(t)$ denote the robot's coordinate at time $t$ in some global coordinate system, and let the robot's orientation (heading direction) be described by $\theta(t)$. The triplet $\langle x, y, \theta \rangle$ describes the kinematic configuration of the robot. The motion of a synchro-drive robot is constrained in a way such that the translational velocity $v$ always leads in the steering direction $\theta$ of the robot, which is a non-holonomic constraint [10]. Let $x(t_0)$ and $x(t_n)$ denote the $x$-coordinates of the robot at time $t_0$ and $t_n$, respectively. Let $v(t)$ denote the translational velocity of the robot at time $t$, and $\omega(t)$ its rotational velocity. Then $x(t_n)$ and $y(t_n)$ can be expressed as a function of $x(t_0)$, $v(t)$ and $\theta(t)$:

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\, dt \tag{1}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\, dt \tag{2}$$

Equations (1) and (2) depend on the velocities of the robot, which usually cannot be set directly. Instead, the velocity $v(t)$ depends on the initial translational velocity $v(t_0)$ at $t_0$, and the translational acceleration $\dot{v}(\hat{t})$ in the time interval $\hat{t} \in [t_0, t]$. Likewise, the orientation $\theta(t)$ is a function of the initial orientation $\theta(t_0)$, the initial rotational velocity $\omega(t_0)$ at $t_0$, and the rotational acceleration $\dot{\omega}(\hat{t})$ with $\hat{t} \in [t_0, t]$. Substituting $v(t)$ and $\theta(t)$ by the corresponding initial kinematic and dynamic configuration $v(t_0), \theta(t_0), \omega(t_0)$ and the accelerations $\dot{v}(\hat{t})$ and $\dot{\omega}(\hat{t})$ yields the expression[1] :

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} \left( v(t_0) + \int_{t_0}^{t} \dot{v}(\hat{t})\, d\hat{t} \right) \cdot \cos\left( \theta(t_0) + \int_{t_0}^{t} \left( \omega(t_0) + \int_{t_0}^{\hat{t}} \dot{\omega}(\tilde{t})\, d\tilde{t} \right) d\hat{t} \right) dt \tag{3}$$

The equations are now in the form that the trajectory of the robot depends exclusively on its initial dynamic configuration at time $t_0$ and the accelerations, which we assume

[1]Notice that the derivation of $y(t_n)$ is analogous, thus we only describe the derivation for the x-coordinate.

to be controllable (for most mobile robots the accelerations determining its motion are monotonic functions of the currents flowing through the motors [6]. Hence, limits on the currents directly correspond to limits on the accelerations).

Digital hardware imposes constraints as to when one can set the motor currents (and thus set new accelerations). Hence, Eq. (3) can be simplified by assuming that between two arbitrary points in time, $t_0$ and $t_n$, the robot can only be controlled by finitely many acceleration commands. Let $n$ denote this number of time ticks. Then, the accelerations $\dot{v}_i$ and $\dot{\omega}_i$ for $i = 1 \dots n$ are kept constant in a time interval $[t_i, t_{i+1}]$ ($i = 1 \dots n$). Let $\Delta_t^i$ and $\Delta_{\hat{t}}^i$ be defined as $\Delta_t^i = t - t_i$ and $\Delta_{\hat{t}}^i = \hat{t} - t_i$ for some interval index $i = 1 \dots n$. Using this discrete form, the dynamic behavior of a synchro-drive robot is expressed by the following equation, which follows directly from Eq. (3) under the assumption of piecewise constant accelerations:

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \left( v(t_i) + \dot{v}_i \cdot \Delta_t^i \right) \cdot \cos\left( \theta(t_i) + \omega(t_i) \cdot \Delta_t^i + \frac{1}{2} \dot{\omega}_i \cdot (\Delta_t^i)^2 \right) \, dt \qquad (4)$$

## 3.2 Approximate Motion Equations

While Eq. (4) describes the general case of mobile root control, it is not particularly helpful when determining the actual steering direction. This is because the trajectories generated by these equations are complex, and geometric operations such as checks for intersections are expensive to perform.

To derive a more practical model, we will now simplify Eq. (4) by approximating the robot's velocities within a time interval $[t_i, t_{i+1}]$ by a constant value. The resulting motion equation, Eq. (6), converges to Eq. (4) as the length of the time intervals goes to zero. As we will see under this assumption the trajectory of a robot can be approximated by piecewise circular arcs. This representation is well-suited for generating motion control in real time, as described in Section 4

If the time intervals $[t_i, t_{i+1}]$ are sufficiently small, the term $v(t_i) + \dot{v}_i \cdot \Delta_t^i$ can be approximated by an arbitrary translational velocity $v_i \in [v(t_i), v(t_{i+1})]$, due to the smoothness of robot motion in time. Likewise, the term $\theta(t_i) + \omega(t_i) + \frac{1}{2}\dot{\omega}_i (\Delta_t^i)^2$ can be approximated by an arbitrary $\theta(t_i) + \omega_i \cdot \Delta_t^i$, where $\omega_i \in [\omega(t_i), \omega(t_{i+1})]$. This leads to the following motion equation

$$x(t_n) \quad = \quad x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cdot \cos\left(\theta(t_i) + \omega_i \cdot (\hat{t} - t_i)\right) d\hat{t} \qquad (5)$$

which, by solving the integral, can be simplified to

$$x(t_n) \quad = \quad x(t_0) + \sum_{i=0}^{n-1} \left( F_x^i(t_{i+1}) \right) \qquad (6)$$

where

$$F_x^i(t) \quad = \quad \begin{cases} \frac{v_i}{\omega_i}(\sin\theta(t_i) - \sin(\theta(t_i) + \omega_i \cdot (t - t_i))), \; \omega_i \neq 0 \\ v_i \cos(\theta(t_i)) \cdot t, \; \omega_i = 0 \end{cases} \qquad (7)$$

The corresponding equations for the $y$-coordinate are:

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} (F_y^i(t_{i+1})) \tag{8}$$

$$F_y^i(t) = \begin{cases} -\frac{v_i}{\omega_i}(\cos\theta(t_i) - \cos(\theta(t_i) + \omega_i \cdot (t - t_i))),\ \omega_i \neq 0 \\ v_i \sin(\theta(t_i)) \cdot t,\ \omega_i = 0 \end{cases} \tag{9}$$

Notice that if $\omega_i = 0$, the robot will follow a straight line. Conversely, if $\omega_i \neq 0$, the robot's trajectory describes a circle, as can be seen by considering

$$M_x^i = -\frac{v_i}{\omega_i} \cdot \sin\theta(t_i) \tag{10}$$

$$M_y^i = \frac{v_i}{\omega_i} \cdot \cos\theta(t_i) \tag{11}$$

for which the following relation holds:

$$\left(F_x^i - M_x^i\right)^2 + \left(F_y^i - M_y^i\right)^2 = \left(\frac{v_i}{\omega_i}\right)^2 \tag{12}$$

This shows that the $i$-th trajectory is a circle $M_i$ about $(M_x^i, M_y^i)$ with radius $M_r^i = \frac{v_i}{\omega_i}$. Hence, by assuming piecewise constant velocities, we can approximate the trajectory of a robot by a sequence of circular and straight line arcs.

Notice that, apart from the initial conditions, Equations (5) to (9) depend only on velocities. When controlling the robot, however, one is not free to set arbitrary velocities, since the dynamic constraints of the robot impose bounds on the maximum deviation of velocity values in subsequent intervals.

## 3.3 An Upper Bound on the Approximation Error

Obviously, the derivation makes the approximate assumption that velocities are piecewise constant within a time interval. This error is bounded linearly in time between control points, $t_{i+1} - t_i$ — a fact which will be used below for modeling uncertainty in the robot's position.

Consider the errors $E_x^i$ and $E_y^i$ for the $x$- and $y$-coordinate, respectively, within the time interval $[t_i, t_{i+1}]$. Let $\Delta t_i := t_{i+1} - t_i$. The deviation in the direction of any of the two axes is maximal if the robot moves on a straight trajectory parallel to that axis. Since in each time interval we approximate $v(t)$ by an arbitrary velocity $v_i \in [v(t_i), v(t_{i+1})]$, an upper bound of the errors $E_x^i$ and $E_y^i$ for $(i+1)$-th time interval is governed by $E_x^i, E_y^i \leq |v(t_{i+1}) - v(t_i)| \cdot \Delta t_i$, which is linear in $\Delta t_i$.

The reader should notice that this bound applies only to the internal prediction of the robot's position. When executing control, the location of the robot is measured periodically with its wheel-encoders (four times a second in our implementation).

This completes the derivation of the robot motion. To summarise, we have derived an approximate form that describes trajectories by sequences of circular arcs, and we have derived a linear bound on the error due to an approximate assumption made in the derivation (piecewise constant velocities).

# 4 The Dynamic Window Approach

In the dynamic window approach the search for commands controling the robot is carried out directly in the space of 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 Figure 3. In the current implementation such a cycle is performed every 0.25 seconds.

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.

## 4.1 Search Space

### Circular trajectories

In Section 3 we showed that it is possible 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 the velocity vector $(v_i,\ \omega_i)$, which we will simply refer as velocity. To generate a trajectory to a given goal point for the next $n$ time intervals the robot has to determine velocities $(v_i,\ \omega_i)$, one for each of the $n$ intervals between $t_0$ and $t_n$. This has to be done under the premise that the resulting trajectory does not intersect with an obstacle. The search space for these vectors is exponential in the number of the considered intervals.

To make the optimization feasible, the dynamic window approach considers exclusively the *first* time interval, and assumes that the velocities in the remaining $n-1$ time intervals are constant (which is equivalent to assuming zero accelerations in $[t_1, t_n]$). 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.

### 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 $(v, \omega)$ the term $dist(v, \omega)$ represents the distance to the closest obstacle on the corresponding curvature (in Section 5.2 we describe how to compute this distance given circular trajectories). A velocity is considered admissible, if the robot is able to stop before it reaches this obstacle. Let $\dot{v}_b$ and $\dot{\omega}_b$ be the accelerations for breakage. Then the set $V_a$ of

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 $(v, \omega)$ of translational and rotational velocities. This results in a two-dimensional velocity search space.
   (b) **Admissible velocities:** The restriction to admissible velocities ensures that only safe trajectories are considered. A pair $(v, \omega)$ is considered admissible, if the robot is able to stop before it reaches the closest obstacle on the corresponding curvature.
   (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

$$G(v, \omega) \;=\; \sigma(\alpha \cdot \mathrm{heading}(v, \omega) + \beta \cdot \mathrm{dist}(v, \omega) + \gamma \cdot \mathrm{vel}(v, \omega)) \tag{13}$$

   is maximized. With respect to the current position and orientation of the robot this function trades off the following aspects:
   (a) **Target heading:** *heading* 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 $\sigma$ smoothes the weighted sum of the three components and results in more side-clearance from obstacles.

Figure 3: Different parts of the dynamic window approach

admissible velocities is defined as

$$V_a = \left\{ (v, \omega) \mid v \leq \sqrt{2 \cdot \mathrm{dist}(v, \omega) \cdot \dot{v}_b} \;\wedge\; \omega \leq \sqrt{2 \cdot \mathrm{dist}(v, \omega) \cdot \dot{\omega}_b} \right\}. \tag{14}$$

Thus $V_a$ is the set of velocities $(v, \omega)$ which allow the robot to stop without colliding with an obstacle.

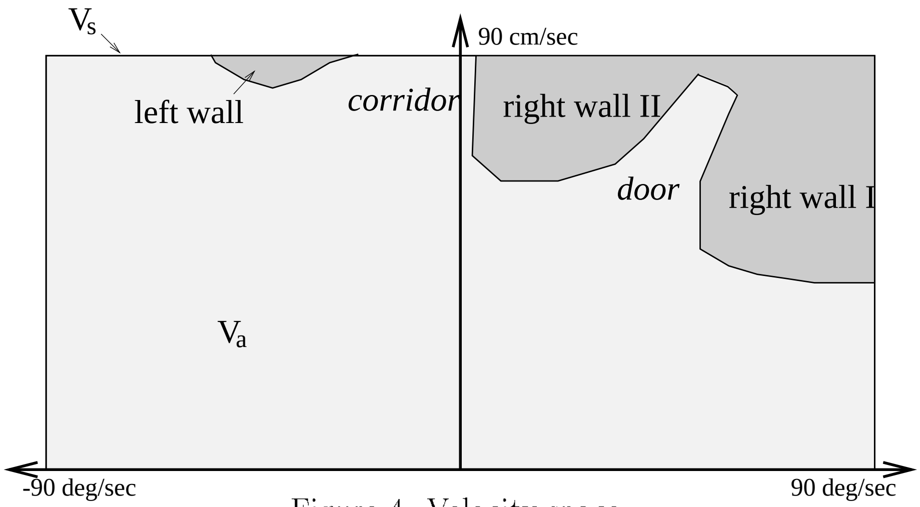

Figure 4. Velocity space

**Example 1** *Again consider the example given in Figure 2. Figure 4 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 extracted from real world proximity information; in this special case this information was obtained from sonar sensors (see Section 5).*

## Dynamic window

In order to take into account the limited accelerations exertable by the motors the overall search space is reduced to the dynamic window which contains only the velocities that can be reached 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 $(v_a, \omega_a)$ be the actual velocity. Then the dynamic window $V_d$ is defined as

$$V_d = \{(v, \omega) \mid v \epsilon [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge \omega \epsilon [\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t]\}\,. \tag{15}$$

The dynamic window is centred 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.

**Example 2** *An exemplary dynamic window obtained in the situation shown in Figure 2 given accelerations of 50 cm/sec*$^2$ *and 60 deg/sec*$^2$ *and a time interval of 0.25 sec is shown in Figure 5. The two dotted arrows pointing to the corners of the rectangle denote the most extreme curvatures that can be reached.*

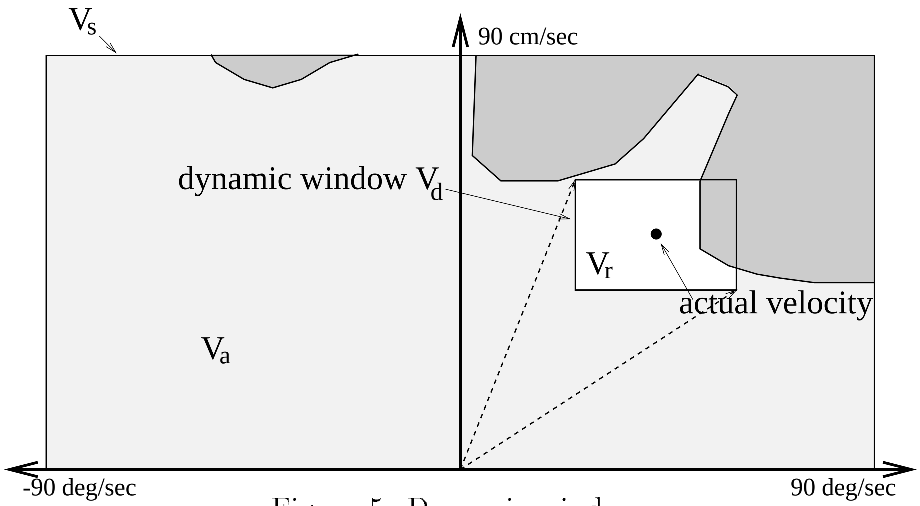

Figure 5. Dynamic window

### Resulting Search Space

The above given restrictions imposed on the search space for the velocities result in the area $V_r$ within the dynamic window. Let $V_s$ be the space of possible velocities, then the area $V_r$ is defined as the intersection of the restricted areas, namely

$$V_r = V_s \cap V_a \cap V_d \tag{16}$$

In Figure 5 the resulting search space is represented by the white area.

## 4.2 Maximizing the Objective Function

After having determined the resulting search space $V_r$ a velocity is selected from $V_r$. In order to incorporate the criteria *target heading*, *clearance*, and *velocity*, the maximum of the objective function

$$G(v, \omega) = \sigma(\alpha \cdot \mathrm{heading}(v, \omega) + \beta \cdot \mathrm{dist}(v, \omega) + \gamma \cdot \mathrm{velocity}(v, \omega))$$

is computed over $V_r$. This is done by discretization of the resulting search space.

### Target heading

The target heading $\mathit{heading}(v, \omega)$ measures the alignment of the robot with the target direction. It is given by $180 - \theta$, where $\theta$ is the angle of the target point relative to the robot's heading direction (see Figure 6). Since this direction changes with the different velocities, $\theta$ is computed for a predicted position of the robot. To determine the predicted position we assume that the robot moves with the selected velocity during the next time interval. For a realistic measurement of the target heading we have to consider the dynamics of the rotation. Therefore, $\theta$ is computed at the position, which the robot will reach when

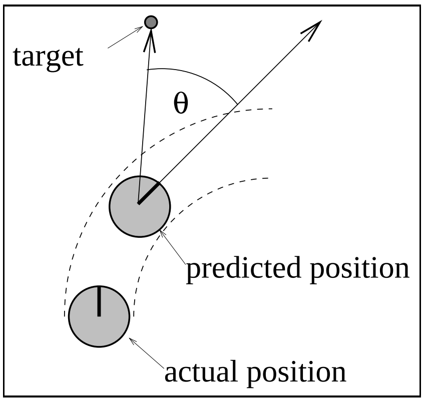


Figure 6. Angle $\theta$ to the target

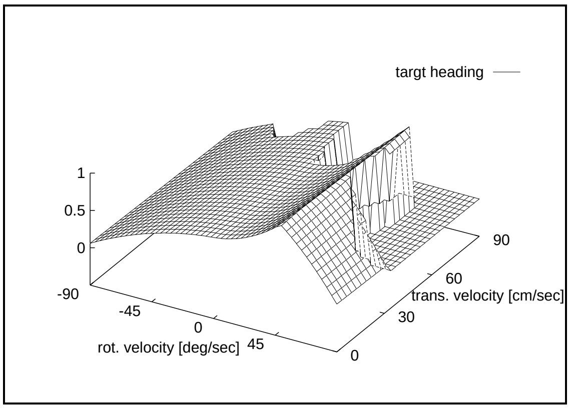


Figure 7. Evaluation of the target heading

exerting maximal deceleration after the next interval. This yields a smooth turning to the target in the behavior of the robot when it has circumvented an obstacle.

**Example 3** *Figure 7 shows the evaluation of the target heading for the different velocities in the example situation. In this figure and the following figures the values for non-admissible velocities are set to zero (compare with Figures 2 and 4). For clarity we show the evaluation of the whole velocity space and do not restrict it to the dynamic window. The non-linearity of the function in Figure 7 is caused by the consideration of the dynamics in the determination of the predicted position. Because positive rotational velocities yield curvatures to the right we find the best velocities on the right side of the velocity space. The optimal velocities are those leading to a perfect heading to the target on the predicted position. The function declines for even higher rotational velocities, because they yield a turning beyond the target.*

### 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.

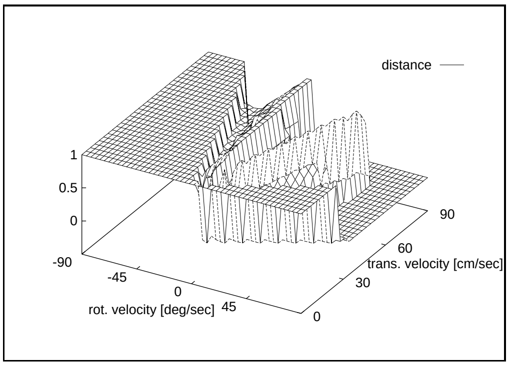


Figure 8. Evaluation of the distances

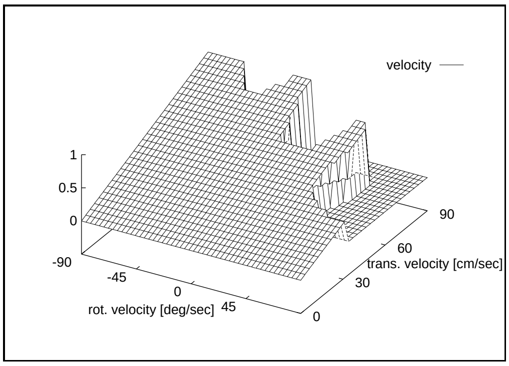


Figure 9. Evaluation of the velocities

**Example 4** *The evaluation of the distances as given in Figure 8 solely depends on the proximity information about obstacles around the robot. In the given plot one can find low evaluations for those curvatures which lead to the walls. For higher translational velocities these values fall into the non-admissible areas and are thus set to zero.*

### 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 $v$, as can be seen in Figure 9.

### Smoothing

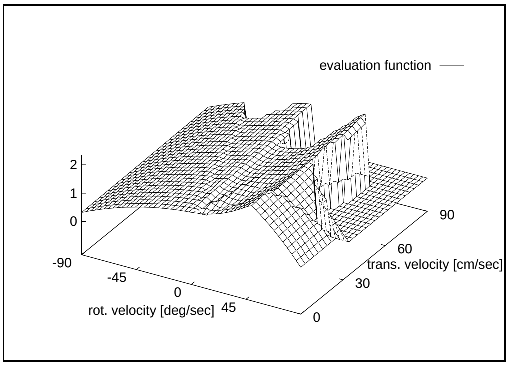


Figure 10. Combined evaluation function

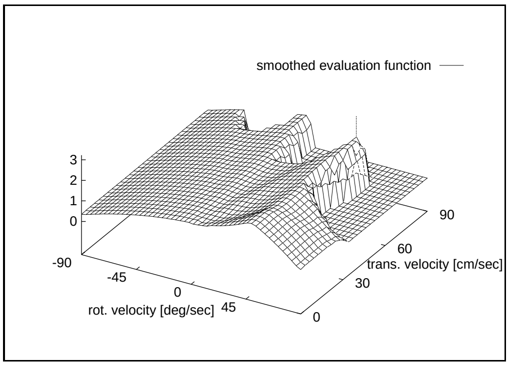


Figure 11. Objective function

All three components of the objective function are normalized to $[0, 1]$. The weighted sum of these components is shown in Figure 10. It is obtained by a value of 2.0 for $\alpha$ and a value of 0.2 for $\beta$ and $\gamma$. As expected the fastest trajectory leading through the door area gets the highest evaluation (compare with Figure 4). Smoothing increases side-clearance of the robot. The resulting objective function is shown in Figure 11 and the position of the maximal value is depicted by the vertical line.

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 circumvents collisions as fast as it can under the constraints listed above, while still making progress towards reaching its goal.

In a former version of our approach (see [3]) the search for the best velocity was carried out in two steps. In the first step only the curvature was chosen. This was done by evaluating the target angle and the so-called "n-sec-rule", namely a linear function of the clearance. In the second step the velocity on this curvature was maximized. Although the resulting behavior of the robot was the same, we decided to use this single step evaluation of the objective function. We adopted the idea for this representation from [13].

## 4.3 Role of the dynamic window

In the previous section we introduced the objective function to be maximized for smooth and goal directed behavior. For illustration purposes we always showed the evaluation for the whole velocity space. As mentioned in Section 4.1 this space is reduced to the admissible velocities in the dynamic window. In this section we describe how the robot respects the dynamics by this restriction. We discuss the dependency of the behavior on different velocities and accelerations. In both examples the time interval determining the dynamic window is fixed to 0.25 seconds.

### Role of the Current Velocity

In this example we show how the behavior changes with the current velocities. We assume accelerations of 50 cm/sec$^2$ and 60 deg/sec$^2$. Figure 12 shows the dynamic windows $V_{d1}$ and $V_{d2}$ given straight motion with translational velocities of 75 and 40 cm/sec, respectively. In the Figures 13 and 14 the objective function is shown for the dynamic windows. Velocities outside the dynamic windows have an evaluation of -1.

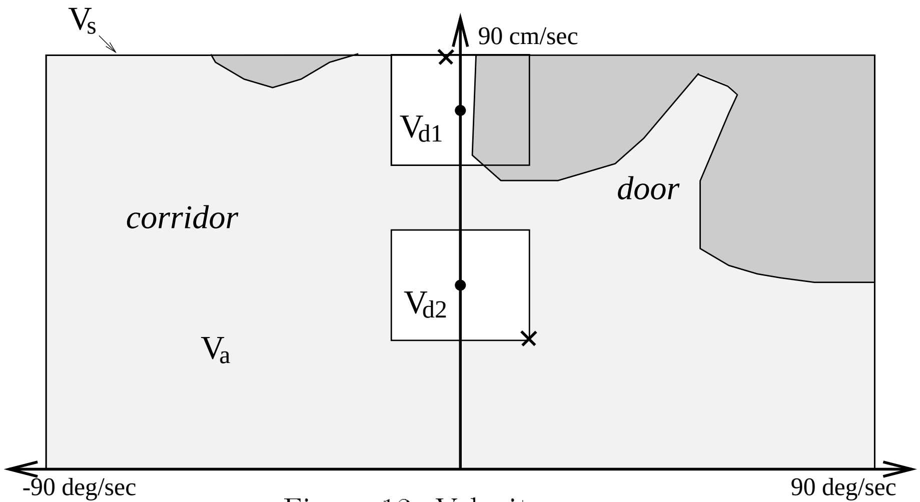

Figure 12. Velocity space

In the first case, where the current velocity is 75 cm/sec, the robot moves too fast to perform the sharp turn to the right necessary to drive through the open door. This is reflected in Figure 13 by the fact that the velocities resulting in a turn to the right are not admissible. Among the velocities in the dynamic window the velocity with maximal evaluation yields straight motion, as denoted by the vertical line in Figure 13 and by the cross mark at the top of $V_{d1}$ in Figure 12.

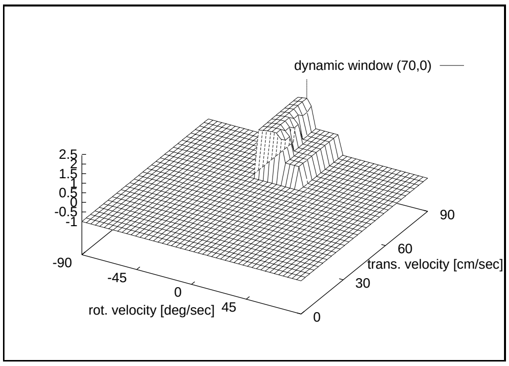


Figure 13. Objective function for actual velocity (75,0)

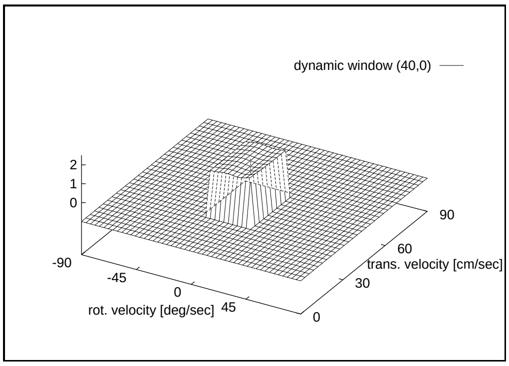


Figure 14. Objective function for actual velocity (40,0)

If in contrast the current velocity is 40 cm/sec, the dynamic window $V_{d2}$ includes curvatures leading through the door (see Figures 12 and 14). Due to the better evaluation of the angle and the distance the chosen velocity is the one yielding the most extreme turn to the right.

**Dependency on the Accelerations**

The restricted search space, the evaluation function, and the dynamic window also depend on the given accelerations. Consider the velocity space for the example with accelerations of 20 cm/sec$^2$ and 30 deg/sec$^2$ as illustrated in Figure 15. Because of the small accelerations the space of admissible velocities is smaller than in Figure 4. Therefore, we only consider velocities up to 60 cm/sec and 50 deg/sec. The dynamic window for straight motion with a translational velocity of 40 cm/sec is represented by the white area. The size of this window is reduced as it depends on the accelerations.

Figure 16 contains the objective function of the entire velocity space. The evaluation of the space restricted to the velocities in the dynamic window is shown in Figure 17. Again the robot is too fast for a sharp turn into the door and the velocity with straight motion has the maximal evaluation (compare to Figure 13).

# 5 Implementation and Experimental Results

## 5.1 RHINO

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.

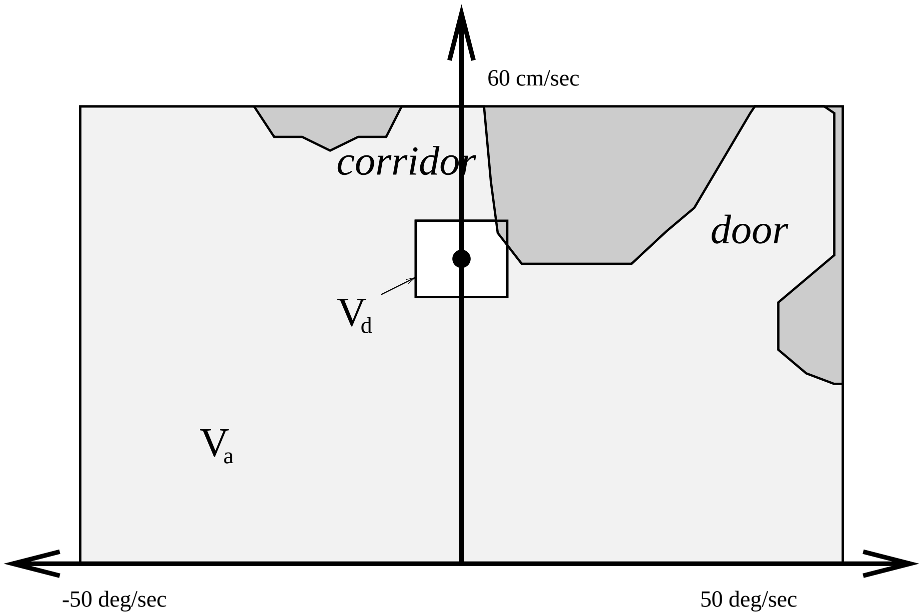

Figure 15. Dynamic window for low accelerations

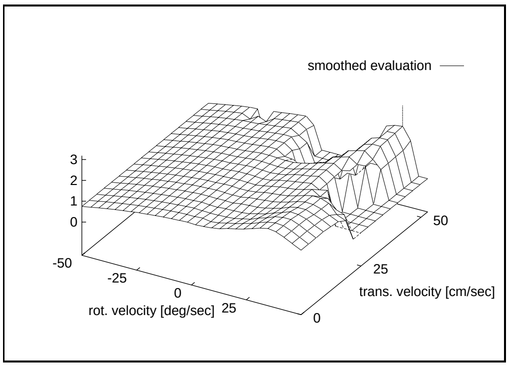

Figure 16. Objective function for low accelerations

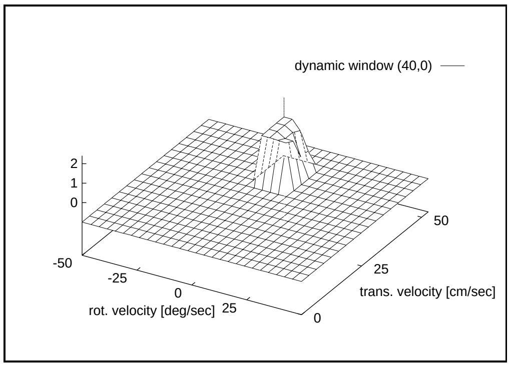

Figure 17. Objective function in dynamic window

## 5.2 The Obstacle Line Field

As local world model we use an obstacle line field [3], which is a two-dimensional description of sensory data relative to the robot's position (see Figure 18). 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 [11] would be required.

The obstacle line field is centred 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

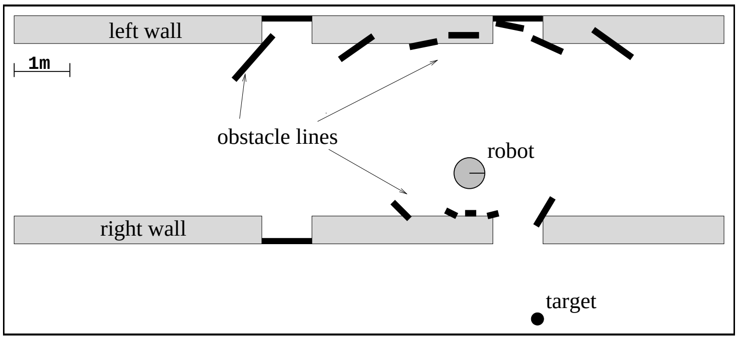


Figure 18. Example environment with obstacle lines and target point

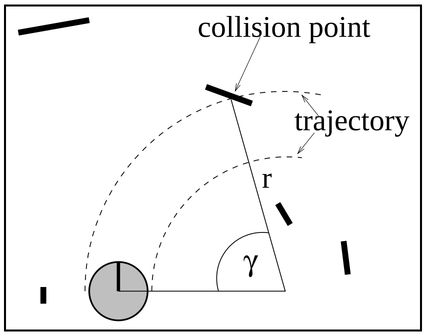


Figure 19. Determination of the distance

this obstacle representation the distance to obstacles on curvatures is computed as follows: let $r$ be the radius of the circular trajectory, and let $\gamma$ be the angle between the intersection with the obstacle line and the position of the robot (see Figure 19). 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.

## 5.3 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 away from the obstacle 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 will travel with high speed through corridors and will decelerate when driving through narrow doors. Simultaneously, possible deviations coming from the approximation error described in Section 3.3 are respected.

## 5.4 Experimental Results

Based on the dynamic window approach to collision avoidance, *RHINO* has been operated safely in various environments, over the last 2 years. Its maximum velocity is constrained by the hardware to approximately 95 cm/sec. *RHINO* reaches this velocity in large openings and hallways, if 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. In the following figures the environment is drawn by hand. Nonetheless each diagram is 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 [14, 15, 16].

### Parameter Settings

Although the performance of the obstacle avoidance depends on the weighting parameters $\alpha$, $\beta$, and $\gamma$, it is stable against slight changes of their values. Without any exhaustive tuning of these parameters we found values of 0.8, 0.1, and 0.1 for $\alpha$, $\beta$, and $\gamma$ to give good results. The alignment of the robot with the target point is mainly determined by the ratio between $\alpha$ and the other two parameters. Very low values of the target heading weight $\alpha$ give the robot much freedom in moving around obstacles. At the same time they may detain the robot from reaching target points behind narrow openings if sensors with low angular resolution are used (e.g. ultrasonic sensors). This is because the robot turns away from the opening before its sensors are able to detect it. On the other extreme, a high heading weight forces the robot to approach objects very closely before turning away, which prohibits smooth circumvention of obstacles.

By choosing different values for $\alpha$ global knowledge about the environment can be transfered to the local obstacle avoidance. While higher values produce good results in narrow environments, smaller values are more appropriate in wide and populated hallways.

### Role of the Dynamic Window

The first experiment demonstrates the influence of the dynamic window on the behavior of the robot. The three paths in Figure 20 are examples for the typical behavior of the robot under different dynamics constraints (for the velocities and accelerations we used the same values as in Section 4.3). 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 a sharp turn 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$(v, \omega)$ becomes very small.

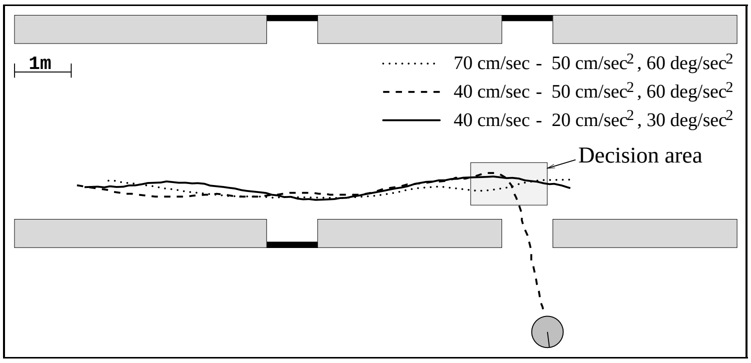


Figure 20. Trajectories chosen for different dynamic parameters

Notice that without considering the dynamic constraints, an attempt to turn right would have almost certainly resulted in a collision with a wall. In fact, in initial experiments with a simulator, in which we ignored some of the dynamic effects, we experienced these type collisions frequently.

### Straight Motion in Corridors

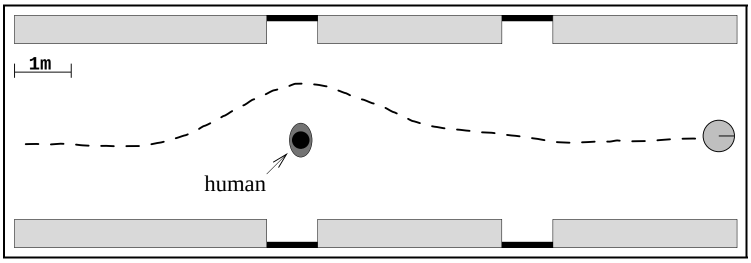


Figure 21. Trajectory through corridor

Figure 21 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 other approaches.

#### Fast Motion through Cluttered Environment

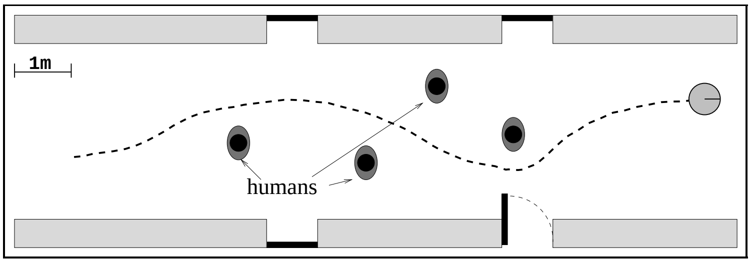


Figure 22. Trajectory through cluttered corridor

The third experiment is shown in Figure 22 and involves traveling through a cluttered corridor. All humans in the corridor are smoothly circumvented with a 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 next trajectory was generated in the arena of the AAAI '94 mobile robot competition. Figure 23 shows a plot of the occupancy map of the arena and the trajectory of the robot. Here the robot moved free of collisions in an artificial indoor environment during an exploration run. 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 21 and 22 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.

# 6 Discussion

This paper describes an approach to collision avoidance for mobile robots equipped with synchro-drives. Based on the exact motion equations of such robots, it derives an approximate version that models robot trajectories by finite sequences of circular arcs. The dynamic window approach first prunes the overall search space by considering only the next steering command. This results in a two-dimensional search space of circular trajectories. After that, the search space is reduced to the admissible velocities allowing the robot

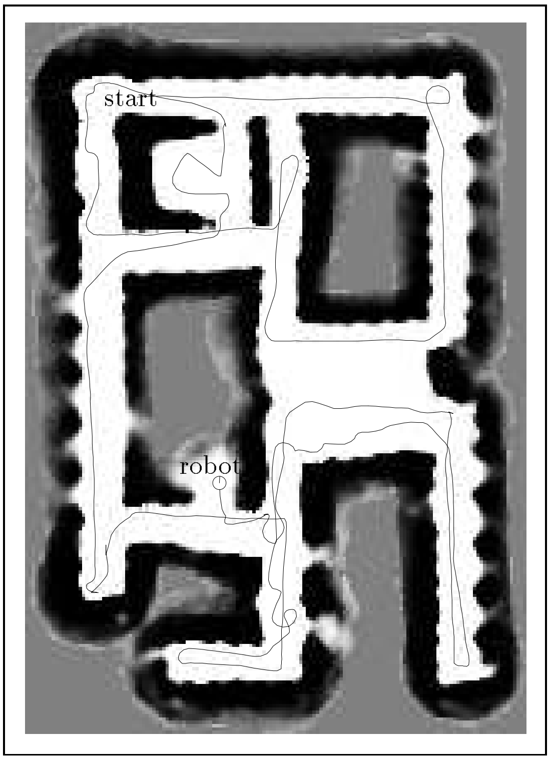


Figure 23. Run at the AAAI '94 mobile robot competition

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 we make sure that the dynamic constraints are taken into account. The robot constantly 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 the objective function.

The experiments show that the combination of all objectives leads to a very robust and elegant collision avoidance strategy which safely operates our robot *RHINO* with speeds of up to 95 cm/sec. *RHINO*, a B21 mobile robot, frequently operates in our university building without human supervision. The approach described here is only part of the overall architecture. For example, approaches to building occupancy maps, global path planning and computer vision are surveyed in [3, 15, 16].

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 such as 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 is converted to proximity information. However, the resulting proximity estimate is only accurate if an obstacle touches the floor. Obstacles in different heights lead to an overestimation of distance, which may 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.

### Acknowledgment

The authors thank Thomas Hofmann and the other members of the RHINO team for insightful discussion, help and advice.

One author (S.T.) is sponsored in part by the National Science Foundation under award IRI-9313367, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Defense Advanced Research Projects Agency (DARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of NSF, Wright Laboratory or the United States Government.

## References

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

[3] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot Rhino. *AI Magazine*, 16(1), 1995.

[4] L. Feng, J. Borenstein, and H.R. Everett. "Where am I?" Sensors and Methods for Autonomous Mobile Robot Positioning. Technical Report UM-MEAM-94-21, University of Michigan, December 1994.

[5] H. Hu and M. 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] J.L. Jones and A.M. Flynn. *Mobile Robots: Inspiration to Implementation.* A K Peters, Ltd., 1993. ISBN 1-56881-011-3.

[7] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. International Conference on Intelligent Autonomous Systems (IAS'4)*, 1995.

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

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

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

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

[12] J.T. Schwartz, M. Scharir, and J. Hopcroft. *Planning, Geometry and Complexity of Robot Motion.* Ablex Publishing Corporation, Norwood, NJ, 1987.

[13] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1996.

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

[15] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI-96*, 1996.

[16] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems.* MIT Press, Cambridge, MA, to appear.