Real-time Obstacle Avoidance for Fast Mobile Robots[1][2]

by
J. Borenstein, Member, IEEE, and Y. Koren, Member, IEEE

ABSTRACT

A new real-time obstacle avoidance approach for mobile robots has been developed and implemented. This approach permits the detection of unknown obstacles simultaneously with the steering of the mobile robot to avoid collisions and advancing toward the target. The novelty of this approach, entitled the Virtual Force Field, lies in the integration of two known concepts: Certainty Grids for obstacle representation, and Potential Fields for navigation. This combination is especially suitable for the accommodation of inaccurate sensor data (such as produced by ultrasonic sensors) as well as for sensor fusion, and enables continuous motion of the robot without stopping in front of obstacles. This navigation algorithm also takes into account the dynamic behavior of a fast mobile robot and solves the "local minimum trap" problem.

Experimental results from a mobile robot running at a maximum speed of 0.78 m/sec demonstrate the power of the proposed algorithm.

---

[1]This work was sponsored (in part) by the Department of Energy Grant DE-FG02-86NE37967

[2]The material in this paper was partially presented at the Third International Symposium on Intelligent Control, Arlington, Virginia, August 24-26, 1988

1. Introduction

Real-time obstacle avoidance is one of the key issues to successful applications of mobile robot systems. All mobile robots feature some kind of collision avoidance, ranging from primitive algorithms that detect an obstacle and stop the robot short of it in order to avoid a collision, through sophisticated algorithms, that enable the robot to detour obstacles. The latter algorithms are much more complex, since they involve not only the detection of an obstacle, but also some kind of quantitative measurements concerning the obstacle's dimensions. Once these have been determined, the obstacle avoidance algorithm needs to steer the robot around the obstacle and resume motion toward the original target.

Autonomous navigation represents a higher level of performance, since it applies obstacle avoidance simultaneously with the robot steering toward a given target. Autonomous navigation, in general, assumes an environment with known and unknown obstacles, and it includes global path planning algorithms [3] to plan the robot's path among the known obstacles, as well as local path planning for real-time obstacle avoidance. This article, however, assumes motion in the presence of unknown obstacles, and therefore concentrates only on the local obstacle avoidance aspect.

One approach to autonomous navigation is the wall-following method [1,17,18]. Here the robot navigation is based on moving alongside walls at a predefined distance. If an obstacle is encountered, the robot regards the obstacle as just another wall, following the obstacle's contour until it may resume its original course. This kind of navigation is technologically less demanding, since one major problem of mobile robots) the determination of their own position) is largely facilitated. Naturally, robot navigation by the wall-following method is less versatile and is suitable only for very specific applications. One recently introduced commercial system uses this method on a floor cleaning robot for long hallways [16].

A more general and commonly employed method for obstacle avoidance is based on edge detection. In this method, the algorithm tries to determine the position of the vertical edges of the obstacle and consequently attempts to steer the robot around either edge. The line connecting the two edges is considered to represent one of the obstacle's boundaries. This method was used in our own previous research [5,6], as well as in several other research projects, such as [9,11,28]. A disadvantage with obstacle avoidance based on edge detecting is the need of the robot to stop in front of an obstacle in order to allow for a more accurate measurement.

A further drawback of edge-detection methods is their sensitivity to sensor accuracy. Unfortunately, ultrasonic sensors, which are mostly used in mobile robot applications, offer many shortcomings in this respect:

1. Poor directionality that limits the accuracy in determination of the spatial position of an edge to 10-50 cm, depending on the distance to the obstacle and the angle between the obstacle surface and the acoustic beam.

2. Frequent misreadings that are caused by either ultrasonic noise from external sources or stray reflections from neighboring sensors ("crosstalk"). Misreadings cannot always be filtered out and they cause the algorithm to "see" nonexisting edges.

3. Specular reflections which occur when the angle between the wave front and the normal to a smooth surface is too large. In this case the surface reflects the incoming ultra-sound waves away from the sensor, and the obstacle is either not detected at all, or (since only part of the surface is detected) "seen" much smaller than it is in reality.

To reduce the effects listed above we have decided to represent obstacles with the Certainty Grid method. This method of obstacle representation allows adding and retrieving data on the fly and enables easy integration of multiple sensors.

The representation of obstacles by certainty levels in a grid model has been suggested by Elfes [15], who used the Certainty Grid for off-line global path planning. Moravec and Elfes [23], and Moravec [24] also describe the use of Certainty Grids for map-building. Since our obstacle avoidance approach makes use of this method, we will briefly describe the basic idea of the Certainty Grid.

In order to create a Certainty Grid, the robot's work area is divided into many square elements (denoted as cells), which form a grid (in our implementation the cell size is 10 cm by 10 cm). Each cell (i,j) contains a Certainty Value $C(i,j)$ that indicates the measure of confidence that an obstacle exists within the cell area. The greater C(i,j), the greater the level of confidence that the cell is occupied by an obstacle.

In our system the ultrasonic sensors are continuously sampled while the robot is moving. If an obstacle produces an echo (within the predefined maximum range limit of 2 meters), the corresponding cell contents C(i,j) are incremented. A solid, motionless obstacle eventually causes a high count in the corresponding cells. Misreadings, on the other hand, occur randomly, and do not cause high counts in any particular cell. This method yields a more reliable obstacle representation in spite of the ultrasonic sensors' inaccuracies.

The novelty of our approach lies in the combination of Certainty Grids for obstacle representation with the Potential Field method for local path planning. Section 2 explains our basic Virtual Force Field (VFF) obstacle avoidance approach in which we apply the Potential Field method to a Certainty Grid. The VFF method is further enhanced by taking into account the dynamic behavior of a mobile robot at high speeds and by a comprehensive heuristic solution to the "trap" problem (which is associated with the Potential Field method). A discussion on these two algorithms is included in Sections 3 and 4. The described algorithms

have been implemented and tested on our mobile robot, CARMEL (**C**omputer-**A**ided **R**obotics for **M**aintenance, **E**mergency, and **L**ife support).

II. The Virtual Force Field (VFF) Method

The idea of having obstacles conceptually exerting forces onto a mobile robot has been suggested by Khatib [20]. Krogh [21] has enhanced this concept further by taking into consideration the robot"s velocity in the vicinity of obstacles. Thorpe [26] has applied the Potential Fields Method to off-line path planning. Krogh and Thorpe [22] suggested a combined method for global and local path planning, which uses Krogh"s Generalized Potential Field (GPF) approach. These methods, however, assume a known and prescribed world model of the obstacles. Furthermore, none of the above methods has been implemented on a mobile robot that uses real sensory data. The closest project to ours is that of Brooks [7,8], who uses a Force Field method in an experimental robot equipped with a ring of 12 ultrasonic sensors. Brooks"s implementation treats each ultrasonic range reading as a repulsive force vector. If the magnitude of the sum of the repulsive forces exceeds a certain threshold, the robot stops, turns into the direction of the resultant force vector, and moves on.

2.1 The Basic VFF Method

This section explains the combination of the Potential Field method with a Certainty Grid. This combination produces a powerful and robust control scheme for mobile robots, denoted as the Virtual Force Field (VFF) method.

As the robot moves around, range readings are taken and projected into the Certainty Grid, as explained above. Simultaneously, the algorithm scans a small square window of the grid. The size of the window is 33x33 cells (i.e., 3.30x3.30m) and its location is such that the robot is always at its center.

Each occupied cell inside the window applies a repulsive force to the robot, "pushing" the robot away from the cell. The magnitude of this force is proportional to the cell contents, $C(i,j)$, and is inversely proportional to the square of the distance between the cell and the robot:

$$\boldsymbol{F}(i,j) = \frac{F_{cr}C(i,j)}{d^2(i,j)}\left[\frac{x_t - x_0}{d(i,j)}\hat{x} + \frac{y_t - y_0}{d(i,j)}\hat{y}\right] \qquad \textbf{(1)}$$

where

F_{cr} = Force constant (repelling)
$d(i,j)$ = Distance between cell (i,j) and the robot
$C(i,j)$ = Certainty level of cell (i,j)
x_0, y_0 = Robot"s present coordinates
x_i, y_j = Coordinates of cell (i,j)

Notice that in Eq. 1 the force constant is divided by d^2. By this method, occupied cells exert strong repulsive forces when they are in the immediate vicinity of the robot, and weak forces when they are further away.

The resultant repulsive force, $\boldsymbol{F}_r$, is the vectorial sum of the individual forces from all cells:

$$\boldsymbol{F}_r = \sum_{i,j} \boldsymbol{F}(i,j) \tag{2}$$

At any time during the motion, a constant-magnitude attracting force, $\boldsymbol{F}_t$, pulls the robot toward the target. $\boldsymbol{F}_t$ is generated by the target point T, whose coordinates are known to the robot. The target-attracting force $\boldsymbol{F}_t$ is given by

$$\boldsymbol{F}(i,j) = F_{ct}\left[\frac{x_t - x_0}{d(t)}\hat{x} + \frac{y_t - y_0}{d(t)}\hat{y}\right] \tag{3}$$

where
F_{ct} = Force constant (attraction to the target)
d_t = Distance between the target and the robot
x_t, y_t = Target coordinates

Notice that $\boldsymbol{F}_t$ is independent of the absolute distance to the target.

As shown in Fig. 1, the vectorial sum of all forces, repulsive from occupied cells and attractive from the target position, produces a resultant force vector $\boldsymbol{R}$:

$$\boldsymbol{R} = \boldsymbol{F}_t + \boldsymbol{F}_r \tag{4}$$

The direction of $\boldsymbol{R}$, $\delta = \boldsymbol{R}/|\boldsymbol{R}|$ (in degrees), is used as the reference for the robot's steering-rate command Ω

$$\Omega = K_s[\theta \text{ (-) } \delta] \tag{5}$$

where
K_s = Proportional constant for steering (in sec^{-1})
θ = Current direction of travel (in degrees)

(-) is a specially defined operator for two operands, ∝ and β (in degrees), and is used in the form
$c = \propto (-) \beta$. The result, c (in degrees), is the shortest rotational difference between ∝ and β. Therefore, c is always in the range $-180^{o} < c < 180^{o}$.

A typical obstacle map in Certainty Grid representation shows obstacle-boundaries as clusters of cells with high certainty values. Misreadings, on the other hand, occur at random and therefore produce mostly isolated cells with low certainty values. Summation of repulsive forces from occupied cells (Eq. 2) makes the robot highly responsive to clusters of filled cells, while almost completely ignoring isolated cells.

2.2 Advantages Over Conventional Methods

The VFF algorithm has several advantages over edge detecting methods, which are presently used in many mobile robot applications:

1. In edge detection methods, misreadings or inaccurate range measurements may be misinterpreted as part of an obstacle contour, thereby gravely distorting the perceived shape of the obstacle. The sharply defined contour required by these methods cannot accommodate the blurry and inaccurate information provided by ultrasonic sensors. Edge detection methods require binary knowledge about the obstacle contour (exists or does not exist), and therefore cannot implement certainty methods, in which the data is weighted. The VFF method, on the other hand, does not utilize sharp contours in the world model, but rather responds to clusters of high likelihood for the existence of an obstacle. This results in

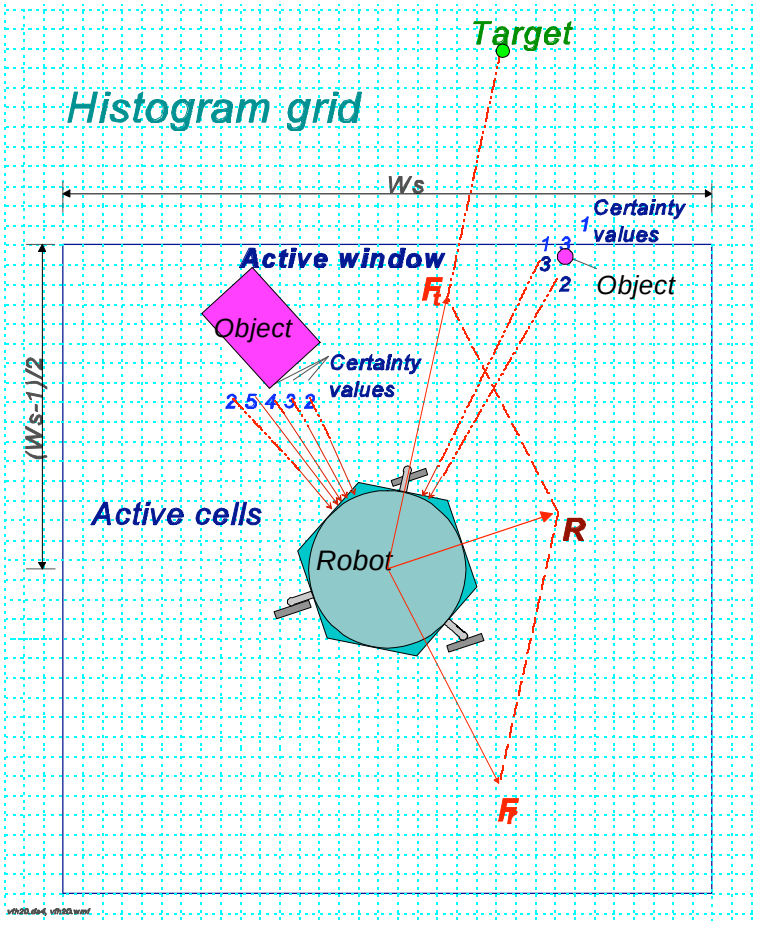

Figure 1: The Virtual Force Field (VFF) concept: Occupied cells exert repulsive forces onto the robot; the magnitude is proportional to the certainty value $c_{i,j}$ of the cell and inversely proportional to d^2.

increased robustness of the algorithm in the presence of misreadings.

2. The VFF method does not require the robot to stop for data acquisition and evaluation, or for corner negotiation (as in the cases reported in [6,8,9,11,12,19]). Except for the artificially introduced effect of damping (see discussion in Section 3.1), the VFF method allows the robot to negotiate all obstacles while it travels at up to its maximum velocity.

3. Updating the grid-map with sensor information and using the grid-map for navigation are two independent tasks that are performed asynchronously, each at its optimal pace. The edge detection method, by contrast, requires the following activities to be performed in sequence: detect an obstacle, stop the robot, measure the obstacle (find its edges), recalculate path, and resume motion.

4. The grid representation for obstacles lends itself easily to the integration of data from groups of similar, as well as from different types of sensors (such as vision, touch, and proximity), in addition to data from previous runs or from preprogrammed stationary obstacles (such as walls).

III. Dynamic Motion Enhancements for Robots Running at High Speeds

This section discusses the main enhancements that have been incorporated in the VFF method in order to account for the dynamic properties of a mobile robot [2,4]moving at higher speeds (up to 0.78 m/sec).

The mobile robot used in all experiments is a commercially available CYBERMATION K2A mobile platform [13]. The K2A has a maximum travel speed of V_{max} = 0.78 m/sec, a maximum
steering rate of Ω=120 deg/sec, and weights (in its current configuration) about W=125 kg. This platform has a unique 3-wheel drive (synchro-drive) that permits omnidirectional steering. In order to control this mobile platform, two data items must be sent (through a serial link) to its onboard computer: a velocity command, V, and a steering-rate command, Ω (computed in Eq. 5 above).

Our mobile platform has been equipped with a ring of 24 Polaroid ultrasonic sensors [25] and an additional computer (a PC-compatible single board computer, running at 7.16 MHz), to control the sensors. Similar sensor configurations have been designed for other mobile robots, such as [14] or [27].

Since the remaining part of this paper focuses on various aspects of **motion performance**, we have chosen to **simulate** obstacles in the Certainty Grid, rather than use real sensory data. The undeterministic nature of actual ultrasonic range information makes it difficult to reproduce test-runs or compare them with each other. The VFF algorithm, however, works

equally well with real obstacles, as is demonstrated in the last experimental result in this paper (Fig. 8).

All of the following examples and plots have been obtained from actual runs of our robot (with its sensors disabled). Although the maximum speed of V_{max}=0.78 m/sec was used, the average speed in each run was lower, since the algorithm reduces the robot"s speed under certain conditions (as will be explained below).

3.1. Low Pass Filter for Steering Control

For smooth operation of the VFF method, the following condition between the grid resolution Δs and the sampling period T must be satisfied:

$$\Delta s > TV_{max} \tag{6}$$

In our case $\Delta s = 0.1$ m and $TV_{max} = 0.1*0.78 = 0.078$ m, and therefore the above condition is satisfied.

Since the distance dependent repulsive force vector $\boldsymbol{F}_r$ (see Eq. 2) is quantized to the grid resolution (10x10 cm), rather drastic changes in the resultant force vector $\underline{R}$ may occur as the robot moves from one cell to another (even with condition (6) satisfied). This results in an overly vivacious steering control, as the robot tries to adjust its direction to the rapidly changing direction of $\boldsymbol{R}$. To avoid this problem, a digital low-pass filter, approximated in the algorithm by $\tau = 0.4$ sec, has been added at the steering-rate command. The resulting steering-rate command is given by

$$\Omega_i = \frac{T\Omega'_{i-1} + (\tau - T)\Omega_{i-1}}{\tau} \tag{7}$$

where
Ω_i = Steering-rate command to the robot (after low-pass filtering)
Ω_{i-1} = Previous steering-rate command
Ω'_i = Steering-rate command (before low-pass filtering)
T = Sampling time (here: $T = 0.1$ sec)
τ = Time constant of the low pass filter

The filter smoothes the robot"s motion when moving alongside obstacles. However, it also introduces a time delay τ, which deteriorates the steering response of the robot by the displacement delay $\tau\Omega$.

3.2 Damping (for Speed Command)

Ideally, when the robot encounters an obstacle, it would move smoothly alongside the obstacle until it can turn again toward the target. At higher speeds (e.g., $V > 0.5$ m/sec), however, the robot introduces a considerable relative delay in responding to changes in steering commands caused by the combined effects of its inertia and the low-pass filter mentioned above. Due to this delay, the robot might approach an obstacle very closely, even if the algorithm produces very strong repulsive forces. When the robot finally turns around to face away from the obstacle, it will depart more than necessary, for the same reason. The resulting path is highly oscillatory, as shown in Fig. 2a.

One way to dampen this oscillatory motion is to increase the strength of the repulsive forces when the robot moves toward an obstacle, and reduce it when the robot moves alongside the obstacle. The general methodology calls for variations in the magnitude of the sum of the repulsive forces $\boldsymbol{F}_r$ as a function of the relative directions of $\boldsymbol{F}_r$ and the velocity vector $\boldsymbol{V}$. Mathematically, this is achieved by multiplying the sum of the repulsive forces by the directional cosine ($\cos\theta$) of the two vectors, $\boldsymbol{F}_r$ and $\boldsymbol{V}$, and using the product as follows:

$$\boldsymbol{F}'_r = w\boldsymbol{F}_r + (1\text{-}w)\,\boldsymbol{F}_r(-\cos\theta) \qquad (8)$$

where $\boldsymbol{F}'_r$ is the **adjusted** sum of the repulsive forces and w is a weighting factor that was set to $w = 0.25$ in our system.

The directional cosine in Eq. 8 is computed by

$$\cos\theta = \frac{V_x F_{rx} + V_y F_{ry}}{|\boldsymbol{V}||\boldsymbol{F}_r|} \qquad \textbf{(9)}$$

where

V_x, V_y = x and y components of velocity vector $\boldsymbol{V}$

F_{rx}, F_{ry} = x and y components of the sum of the repulsive forces, $\boldsymbol{F}_r$

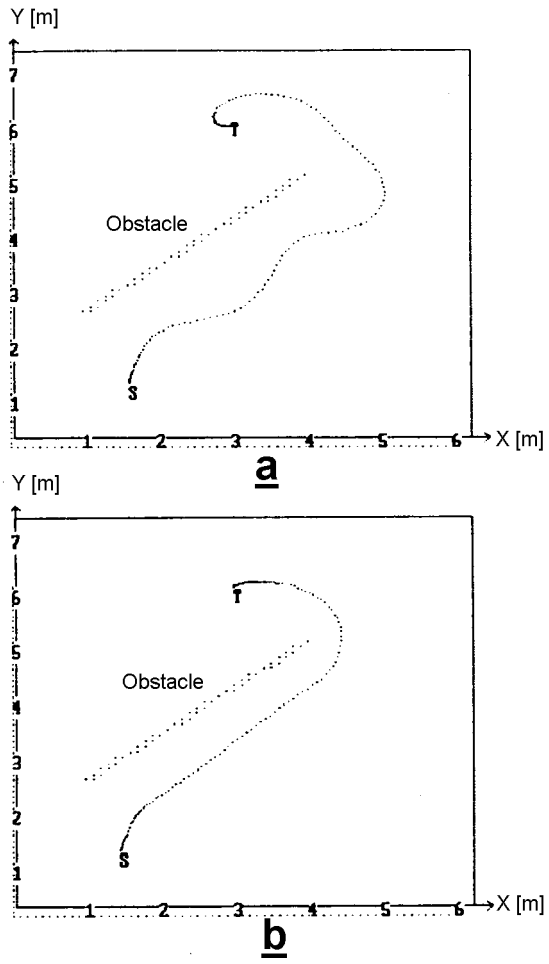

Figure 2:
a. Highly oscillatory motion in undamped force field.
b.The robot"s path is effectively smoothed as force and speed damping are applied.

The effect of this damping method is that the robot experiences the repulsive forces at their full magnitude, as it approaches the obstacle frontally (with $-\cos\theta=1$). As the robot turns toward a direction alongside the obstacle"s boundary, the repulsive forces are weakened by the factor $0.75*\cos\theta$, and will be at their minimum value when the robot runs parallel to the boundary. Notice that setting $w=0$ is undesirable, since the robot will eventually run into an obstacle as it approaches it at a very small angle.

Careful examination of Eq. 8 reveals the fact that the damped sum of repulsive forces, $\boldsymbol{F}'_r$, may become negative (thereby actually attracting the robot), as the robot moves away from the obstacle (and $\cos\theta>0$). We found the attraction-effect to improve damping and reduce oscillatory motion.

3.3 Speed Control

The intuitive way to control the speed of a mobile robot in the VFF environment is to set it proportional to the magnitude of the sum of all forces, $\boldsymbol{R} = \boldsymbol{F}_r + \boldsymbol{F}_t$. Thus, if the path was clear, the robot would be subjected only to the target force and would move toward the target, at its maximum speed. Repulsive forces from obstacles, naturally opposed to the direction of $\underline{F}_t$ (with disregard to the damping effect discussed above), would reduce the magnitude of the resultant $\underline{R}$, thereby effectively reducing the robot"s speed in the presence of obstacles.

However, we have found that the overall performance can be substantially improved by setting the speed command Ω proportional to $\cos\theta$ (see Eq. 9). This function is given by:

$$V = \begin{cases} V_{\max} & \text{for } |\boldsymbol{F}_r| = 0 \quad \text{(i.e. in the absence of obstacles)} \\ V_{\max}(1 - |\cos\theta|) & \text{for } |\boldsymbol{F}_r| > 0 \end{cases} \qquad \textbf{(10)}$$

With this function, the robot still runs at its maximum speed if no obstacles are present. However, in the presence of obstacles, speed is reduced only if the robot is heading toward the obstacle (or away from it), thus creating an additional damping effect. If, however, the robot moved alongside an obstacle boundary, its speed is almost not reduced at all and it moves at its maximum speed, thereby greatly reducing the overall travel-time.

Fig. 2b shows the joint effect of both damping measures on the resulting path.

IV. Recovery from "Local Minimum Traps"

One problem inherent to the basic VFF method is the possibility for the robot to get "trapped." This situation may occur when the robot runs into a dead end (e.g., inside a U-shaped obstacle). Traps can be created by a variety of different obstacle configurations, and different types of traps can be distinguished. This section presents a comprehensive set of heuristic rules to recover from different trap conditions. Chattergy [10] presented some heuristic local path planning solutions for various obstacle configurations (and trap conditions), based on distance measurements to the obstacle. While his approach to recovery from a **single** trap is similar to ours (through wall-following, see discussion below), his solution to the problem of multiple traps differs completely from ours. Also, Chattergy offers no solution to the inside-wall problem (as discussed in section 4.2).

4.1 Trap-state Detection

In an ideal, non-inertial system, trap-states may be discovered by simply monitoring the speed of the robot. If caught in a trap, the robot"s speed will become zero as the robot converges to the equilibrium position with $\boldsymbol{R} = 0$. In a dynamic system, however, the robot overshoots the equilibrium position and will either oscillate or run in a closed loop, as shown in Fig. 3a for an actual run. Therefore, it is impractical to monitor the magnitude of the resultant force $|\boldsymbol{R}|$ for trap-state detection. Our method for trap-state detection compares the Robot-To-Target direction, θ_t, with the actual instantaneous direction of travel, θ_0. If the robot"s direction of travel is more than 90° off-target, namely, if

$$|\theta_t - \theta_0| > 90^\circ \qquad (12)$$

the robot starts to move away from the target and is very likely about to get trapped. Therefore, to avoid trap-situations, the controller monitors the condition in Eq. 12. If Eq. 12 is satisfied, the system switches to the recovery algorithm discussed below. Notice that Eq. 12 expresses an over-conservative condition, and under certain circumstances this condition may be satisfied without the robot being subsequently trapped.

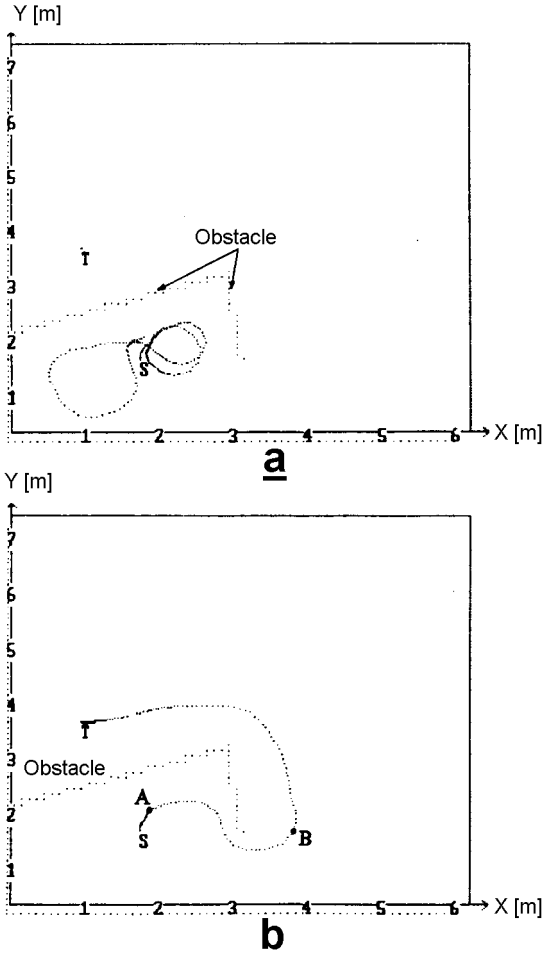

Figure 3:
a. The robot is caught in a "local minimum trap".
b. As the robot finds itself 90° off target, it goes into wall-following mode (at "A") and reaims at the target (at "B").

However, this test is computationally simple, and the evoked recovery algorithm does not reduce the overall performance (if evoked only temporarily), even if the robot would not be trapped.

4.2 Recovery Through Wall-following

The recovery algorithm, referred to as the wall-following method (WFM), steers the robot such as to follow the current obstacle contour. The current obstacle contour is the boundary of the obstacle that "pushed" the robot away and made it assume a heading more than 90° off the target direction. With the WFM, the robot follows the contour until it starts heading back into a direction less than 90° off-target (i.e., $|\theta_t - \theta_0| < 90°$). Subsequently, the robot resumes its (normal) VFF mode, and heads toward the target. The joint operation of the WFM and the VFF mode is demonstrated in Fig. 3b, in which the robot reaches the target in 16 sec at an average speed of 0.44 m/sec. At point A, the algorithm detects a trap situation, and switches into WFM. At point B the off-angle becomes less than 90°, and the algorithm switches back into VFF mode. This recovery algorithm works even for extreme constellations of obstacles, such as the labyrinth-like obstacle course in Fig. 4. Again, at point A the system activates the WFM algorithm, and at point B it switches back to VFF mode.

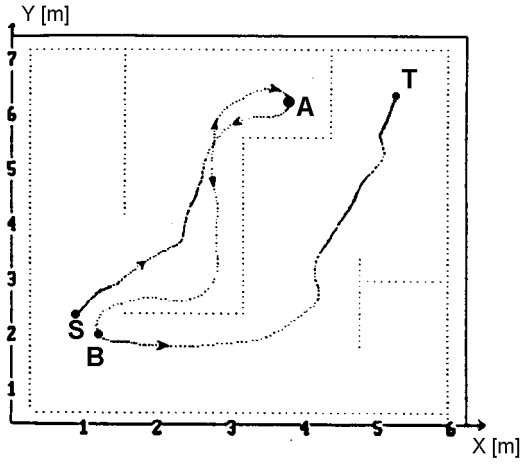

Figure 4: Robot successfully recovers from a labyrinth-like obstacle course.

The WFM algorithm is implemented in the following manner: First, the robot calculates the sum of all repulsive forces, $\underline{F}_r$, in order to determine the direction of $\underline{F}_r$, θ_r. Next, as is depicted in Fig. 5 (for following a wall to the left of the robot), the algorithm adds an angle $\propto$ to θ_r, where $90° < \propto < 180°$ (or subtracts $\propto$ from θ_r, if following a wall to the right of the robot). In our system $\propto = 145°$. Then, the algorithm projects a new virtual attractive force $\underline{F}''_t$ in the resulting direction, which temporarily replaces the attractive force from the target location, $\underline{F}_0$. The new resultant $\underline{R}$ will now point (or eventually converge) into a direction parallel to the obstacle boundary. While wall-following could also be implemented through controlling a fixed distance to the wall, our method is preferable,

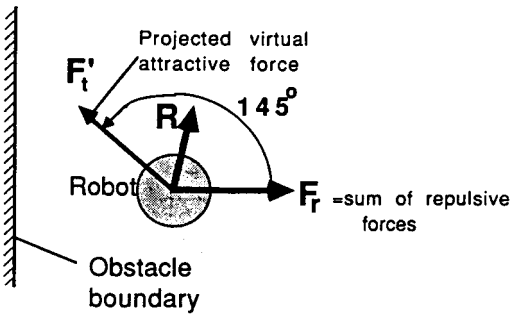

Figure 5: Derivation of the virtual attractive force $\underline{F}''_t$ for the wall-following mode.

since it is less sensitive to misreadings of the ultrasonic sensors. A similar implementation for wall-following has been suggested by Brooks and Connell [8], with $\propto = 120^o$.

There is, however, one catch to the WFM, which may occur if more than one trap is present. Figure 6a shows one such case. Here the robot switches to WFM at point A and follows a wall to its left. Then, at point B, it switches back to VFF mode, since its heading is now less than 90° off the target direction. At point C, a new trap-condition is detected, and the robot goes again into WFM (notice that at this time the robot follows the obstacle at its right).
This pattern repeats itself at points D, E, and F, inside the second and third trap. Subsequently, the robot returns and oscillates between the traps. To solve this problem, two possible wall-following modes must be distinguished: L-mode, where the robot follows a wall to its left, and R-mode, where the robot follows a wall to its right. This distinction is utilized in the implementation of the WFM in our system, as is explained below.

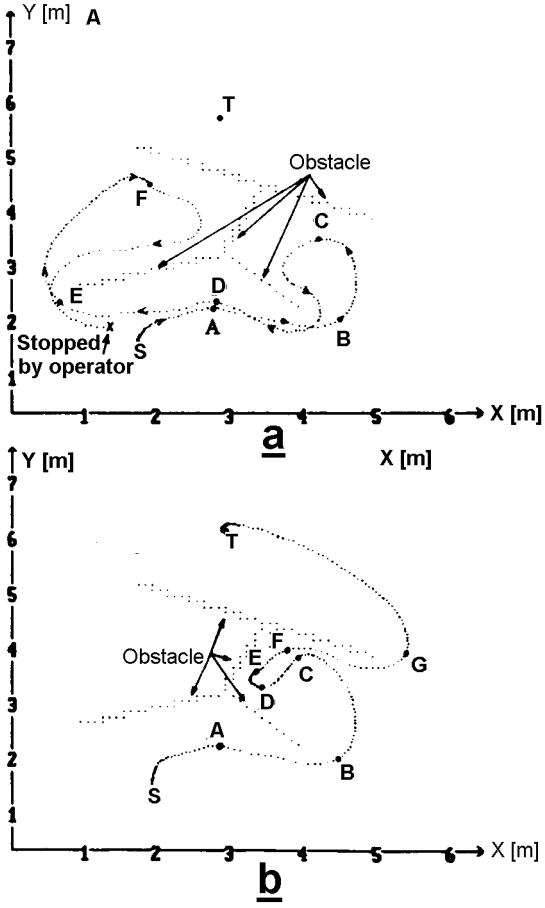

Figure 6:
a. Robot oscillates between multiple traps.
b. Remedy of multiple trap oscillations by adherence to original wall-following mode.

In a given run the robot might select either L- or R-mode at the first trap-situation, but then it must stick always to this mode within the given run. The result of running this algorithm is depicted in Fig. 6b (for the same obstacle configuration as in Fig. 6a). The robot encounters a trap at point A and chooses L-mode. At point B the trap-situation is resolved and the robot returns to VFF mode. However, a new obstacle is encountered, which "pushes" the robot into R-mode (at point C). Since the robot has to stick to the original WFM, it slows down and performs a U-turn (at point D). Subsequently, the robot resumes motion in VFF mode (at point E). A new trap is detected at point F, but this trap can be resolved by running in (the original) L-mode. At point G the robot resumes motion in VFF mode and reaches the target. The average speed for this run was 0.5 m/sec.

One last exception that needs to be addressed occurs when the robot is in WFM on an **inside wall** of a closed room (with the target in the same room). In this case, the robot will follow that wall indefinitely, since the condition for exiting WFM ($|\theta_t - \theta_0| < 90^o$) will not be satisfied (Note that θ_0 is updated absolutely and not by modulus 360°, so that, e.g., $420^o \neq 60^o$).

However, the above situation may be detected by monitoring the sum Φ of the changes of the target direction, θ_t, between sampling intervals i.

$$\Phi = \Sigma[\theta_t(i) - \theta_t(i-1)] \quad (13)$$

$\Phi > 360^o$ indicates that the robot has traveled at least one full loop **around** the target. This only happens when the robot follows an inside wall completely surrounding the target.

Once detected, there are several ways to remedy the situation. In our algorithm, the robot is simply forced out of the WFM and back into normal target pursuit. Fig. 7 shows an example: At point A the algorithm detects the trap condition and switches into WFM with R-mode. At point B the robot has completed one full revolution about the target (since A) and the loop condition ($\Phi > 360^o$) is detected. The robot slows down and stops at point C. Theoretically, there is no need to stop at C, but for the trivial purpose of untangling the umbilical cord (note that the robot has spun almost 720^o by the time it reaches C) the robot does stop at C. It then rotates on the spot until its heading is directed toward the target again, and resumes motion in VFF mode.

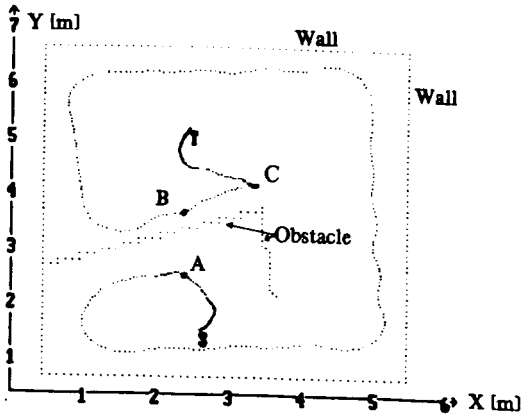

Figure 7: Wall-following on the inner wall of a room may result in an idefinite loop. However, a full loop around the target can be identified and used to remedy the situation.

Fig. 8 shows a run of the robot with **actual ultrasonic data**, obtained in real-time during the robot"s motion. Partitions were set up in the lab such as to resemble the simulated obstacles in Fig. 3. The robot ran at a maximum speed of 0.78 m/sec and achieved an average speed of 0.53 m/sec. The maximal range for the sensors was set to 2 m, which is why only part of the rightmost wall is shown, whereas the rear wall and most of the leftmost wall remained undetected.

Each dot in Fig. 8 represents one cell in the Certainty Grid. In our current implementation, Certainty Values (CVs) range only from 0 to 3. CV = 0 means no sensor reading has been projected into the cell during the run (no dot at all). CV = 1 (or CV = 2) means that one (or two) readings have been projected into the cell, and this is shown in Fig. 8 with dots comprising of 1 (or 2) pixels. CV = 3 means that 3 or more readings have been projected into the same cell, and this is represented by a 4-pixel dot in Fig. 8.

At least two misreadings can be identified in Fig. 8, which have been encircled for emphasis.

V. Conclusions

A comprehensive obstacle avoidance approach for fast-running mobile robots, denoted as the VFF method, has been developed and tested on our experimental mobile robot CARMEL. The VFF method is based on the following principles:

1. A Certainty Grid for representation of (inaccurate) sensory data about obstacles provides a robust real-time world model.

2. A field of virtual attractive and repulsive forces determines the direction and speed of the mobile robot.

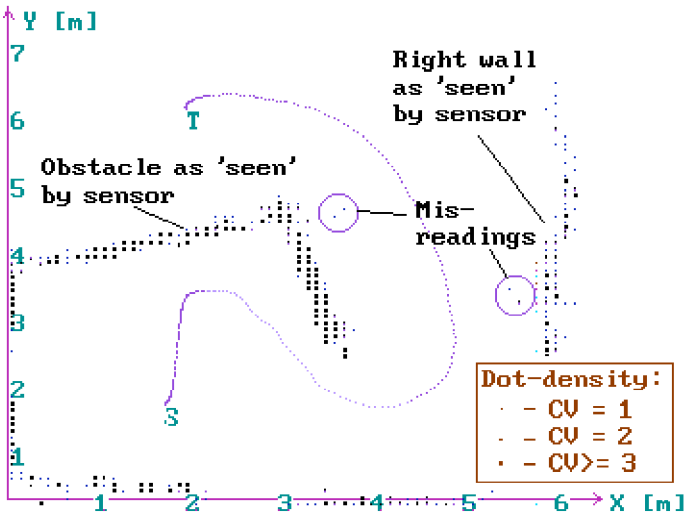

Figure 8: Robot run with **actual ultrasonic data** obtained in real-time during the robot"s motion. Maximum speed = 0.78 m/sec and average speed = 0.53 m/sec.

3. The combination of 1. and 2. results in the characteristic behavior of the robot: The robot responds to clusters of high-likelihood for the existence of an obstacle, while ignoring single (possibly erroneous) data points.

4. Trap states are automatically detected and recovery routines are activated. These routines distinguish among several different situations and take appropriate action for each situation.

5. Oscillatory robot motion is resolved by damping algorithms (which only marginally compromise the robot"s average speed).

Based on the VFF method for autonomous obstacle avoidance, we are currently developing a new mode of operation for the remote guidance of mobile robots. Under this mode of operation, the mobile robot follows the general direction prescribed by the operator. If the robot encounters an obstacle, it autonomously avoids collision with that obstacle, trying to match the prescribed direction as good as possible. With this integrated self-protection mechanism, robots can be steered at high speeds and in cluttered environments, even by inexperienced operators.

IV. References

[1] Bauzil, G., Briot, M. and Ribes, P., "A Navigation Sub-System Using Ultrasonic Sensors for the Mobile Robot HILARE." 1st In.. Conf. on Robot Vision and Sensory Controls, Stratford-upon-Avon, UK., 1981, pp. 47-58 and pp. 681-698.

[2] Borenstein, J. and Koren, Y., "A Mobile Platform For Nursing Robots." IEEE Transactions on Industrial Electronics, Vol. 32, No. 2, 1985, pp. 158-165.

[3] Borenstein, J. and Koren, Y., "Optimal Path Algorithms For Autonomous Vehicles." Proceedings of the 18th CIRP Manufacturing Systems Seminar, June 5-6, 1986, Stuttgart.

[4] Borenstein, J. and Koren, Y., "Motion Control Analysis of a Mobile Robot." Transactions of ASME, Journal of Dynamics, Measurement and Control, Vol. 109, No. 2, 1987, pp. 73-79.

[5] Borenstein, J., "The Nursing Robot System." Ph. D. Thesis, Technion, Haifa, Israel, 1987,.

[6] Borenstein, J. and Koren, Y., "Obstacle Avoidance With Ultrasonic Sensors." IEEE Journal of Robotics and Automation., Vol. RA-4, No. 2, 1988, pp. 213-218.

[7] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot." IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, 1986, pp. 14-23.

[8] Brooks, R. A. and Connell, J. H., "Asynchronous Distributed Control System for a Mobile Robot", Proceedings of the SPIE, Vol. 727, Mobile Robots, 1986, pp. 77-84.

[9] Cooke, R. A., "Microcomputer Control of Free Ranging Robots." Proc. of the 13th Int. Symp. on Industrial Robots and Robots, Chicago, Ill., April, 1983, pp. 13.109-13.120.

[10] Chattergy, R., "Some Heuristics for the Navigation of a Robot"." The International Journal of Robotics Research, Vol. 4, No. 1, 1985, pp. 59-66.

[11] Crowley, J. L., "Dynamic World Modeling for an Intelligent Mobile Robot." IEEE Seventh International Conference on Pattern Recognition, Proceedings July 30-August 2, Montreal, Canada, 1984, pp. 207-210.

[12] Crowley, J. L., "Navigation for an Intelligent Mobile Robot." Carnegie-Mellon University, The Robotics Institute, Technical Report, August, 1984.

[13] Cybermation, "K2A Mobile Platform." Commercial Offer, 5457 JAE Valley Road, Roanoke, Virginia 24014, 1987.

[14] Denning Mobile Robotics, Inc., "Securing the Future." Commercial Offer, 21 Cummings Park, Woburn, MA 01801, 1985.

[15] Elfes, A., "A Sonar-Based Mapping and Navigation System." Carnegie-Mellon University, The Robotics Institute, Technical Report, 1985, pp. 25-30.

[16] Engelberger, J., Transitions Research Corporation, private communication, 1986.

[17] Giralt, G., "Mobile Robots." NATO ASI Series, Vol. F11, Robotics and Artificial Intelligence, Springer-Verlag, 1984, pp. 365-393.

[18] Iijima, J., Yuta, S., and Kanayama, Y., "Elementary Functions of a Self-Contained Robot "YAMABICO 3.1" ." Proc. of the 11th Int. Symp. on Industrial Robots, Tokyo, 1983, pp. 211-218.

[19] Jorgensen, C., Hamel, W., and Weisbin. C. "Autonomous Robot Navigation." BYTE, January, 1986, pp. 223-235.

[20] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." 1985 IEEE International Conference on Robotics and Automation, March 25-28, 1985, St. Louis, pp. 500-505.

[21] Krogh, B. H., "A Generalized Potential Field Approach to Obstacle Avoidance Control." International Robotics Research Conference, Bethlehem, PA, August, 1984.

[22] Krogh, B. H. and Thorpe, C. E., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles." Proceedings of the 1986 IEEE International Conference on
Robotics and Automation, San Francisco, California, April 7-10, 1986, pp. 1664-1669.

[23] Moravec, H. P. and Elfes, A., "High Resolution Maps from Wide Angle Sonar." IEEE Conference on Robotics and Automation, Washington, D.C., 1985, pp. 116-121.

[24] Moravec, H. P., "Certainty Grids for Mobile Robots." Preprint of Carnegie-Mellon University, The Robotics Institute, Technical Report, 1986.

[25] Polaroid Corporation, Ultrasonic Ranging Marketing, 1 Upland Road, MA 02062, 1982.

[26] Thorpe, C. F., "Path Relaxation: Path Planning for a Mobile Robot." Carnegie-Mellon University, The Robotics Institute, Mobile Robots Laboratory, Autonomous Mobile Robots, Annual Report 1985, pp. 39-42.

[27] Walter, S. A., "The Sonar Ring: Obstacle Detection for a Mobile Robot." Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, March 31 - April 3, 1987, pp. 1574-1579.

[28] Weisbin, C. R., de Saussure, G., and Kammer, D., "SELF-CONTROLLED. A Real-Time Expert System for an Autonomous Mobile Robot." Computers in Mechanical Engineering,
September, 1986, pp. 12-19.