Sensory Based Motion Planning with Global Proofs

Ishay Kamon

Computer Science Department
Technion, Technological Institute of Israel
Haifa 32000, Israel

Ehud Rivlin

Computer Science Department
Technion, Technological Institute of Israel
Haifa 32000, Israel

Abstract

A sensory based algorithm DistBug, that is guaranteed to reach the target in an unknown environment or report that the target is unreachable, is presented. The algorithm is reactive in the sense that it relies on range data to make local decisions, and does not create a world model. The algorithm consists of two behaviors (modes of motion): straight motion between obstacles and obstacle boundary following. Simulation results as well as experiments with a real robot are presented.

The condition for leaving obstacle boundary is based on the free range in the direction to the target. This condition allows the robot to leave the obstacle as soon as the local conditions guarantee global convergence. Range data is utilized for choosing the turning direction when the robot approaches an obstacle. A criterion for reversing the boundary following direction when it seems to be the wrong direction is also introduced. As a direct result of these local decisions, a significant improvement in the performance was achieved.

1 Introduction

Finding a path from a given location to a given destination is a fundamental problem in mobile robots research. Traditional methods assume perfect knowledge about the environment. In a more realistic setting the robot should rely on its sensors in order to perceive its environment and plan accordingly. A possible approach is to use the sensors to create a model of the environment (world reconstruction). Traditional path planning methods could then be applied to the world model. Reconstructing a world model based on sensory information is not an easy task because of the inherent uncertainty of the data. Various methods were proposed for data/sensor fusion (see for example [2], [3], [9]).

Sensory-based algorithms use local sensory information for feedback control of the robot motion. This information enables the robot to operate in an unknown environment. At every time instance, the robot uses its sensors to locate close obstacles, and then plans the next part of its path. Reactive strategies emphasize the direct and immediate relations between perception (input) and action (output), and does not rely on world models. Using reactive navigation a robot can adjust its operation in changing and dynamic environments.

Many reactive algorithms for sensory-based path planning are based on the idea of "navigation function" that maps sensor readings to actions. Different methods are used to choose or to learn this mapping, including potential fields and its variations and fuzzy logic (see for example [5],[1],[11],[4]). Local potential field approaches usually suffer from local minima problems. Another problem of the navigation function approaches is falling into a loop. The sensory information is inherently local, and the same action will be always taken for the same local situation, without any global considerations. Therefore once trapped in a loop, there is no way to get out of it.

A different approach, which guarantees reaching the target if possible, was presented by Lumelsky and Stepanov [7]. The robot, equipped only with contact sensors, goes straight to the target until hitting an obstacle. It then follows the object boundary. The algorithms define "leaving conditions" that determine where to leave the obstacle boundary and go directly towards the target again. The convergence is based on the fact that the distance from every leave point to the target is strictly smaller than the distance from the corresponding hit point. Algorithms that use only the direction to the target and abandon obstacles as soon as possible were presented in [10] and [8]. Range sensing was used in [6] to find "shortcuts" to the path that would be planned using contact sensors.

We will present a new algorithm *DistBug* that uses range information more effectively. The main contribution is a new "leaving condition" that allows the robot to abandon obstacles as soon as global convergence is guaranteed, based on the free range in the direction to the target. Range data is utilized for choosing the turning direction when the robot approaches an obstacle. A criterion for reversing the boundary following direction when it seems to be the wrong direction is also introduced. As a direct result of these local decisions, a significant improvement in the performance was achieved.

We consider a point robot that moves among arbitrary obstacles in a planar environment. We assume that the workspace is bounded. It follows directly that the perimeter of any obstacle is finite, and that the number of obstacles is finite.

The rest of the paper is organized as follows: Section 2 presents the algorithm *DistBug* and gives a convergence proof for it. In Section 3 the experimental setting and results are presented. Finally, the conclusions are presented in Section 4.

2 *DistBug* algorithm

The algorithm *DistBug* consists of two modes of motion: moving directly towards the target between obstacles and following obstacle boundaries. The robot goes straight to the target until hitting an obstacle. The Obstacle is then followed until a condition for leaving the obstacle boundary holds. The leaving condition, described in section 2.1, guarantees the convergence of the algorithm. The mechanism for choosing boundary following direction, described in section 2.2, influences the path length. The algorithm is as follows:

1. Go directly to the target, until one of the following occurs:
   a) The target is reached. Stop.
   b) An obstacle is reached. Go to step 2.
2. Choose the boundary following direction. Follow the obstacle boundary until one of the following occurs:
   a) The target is reached. Stop.
   b) The free range in the direction to the target guarantees that the next hit point will be closer to the target than the last hit point.Go to step 1.
   c) The robot completed a loop around the obstacle. The target is unreachable. Stop.

2.1 Leaving boundaries as soon as convergence is guaranteed

Motivated by the fact that straight motion is faster and safer than boundary following, the condition for leaving an obstacle boundary is designed to abandon the boundary as soon as convergence is guaranteed. Range information in the direction to the target is used to guarantee that the next hit point will be closer to the target relative to the last hit point.

We will use the following notations and definitions: $Curr_{dist}$ – distance from the current location to the target; Hit_{dist} – distance from the last hit point H to the target; $Step$ – a constant that will be the minimal improvement in the distance to the target between hit points. $Best_{dist}$ – the up-to-date shortest distance to the target since the last hit point. It is initialized to $Best_{dist} \leftarrow (Hit_{dist} - Step)$. $Free_{dist}$ – distance in freespace from the current location to the nearest obstacle in the direction to the target. We consider a limited sensor range R. If no obstacles are detected we set $Free_{dist} \leftarrow R$;

We define the following leaving condition:

If $(\boldsymbol{Free_{dist} > 0})$ **and**
$((\boldsymbol{Curr_{dist} - Free_{dist} \leq 0})$ **or**
$(\boldsymbol{Curr_{dist} - Free_{dist} \leq Best_{dist}}))$
then leave the obstacle.

The condition $(Free_{dist} > 0)$ checks whether it is possible to leave the obstacle boundary and move directly towards the target. The condition $(Curr_{dist} - Free_{dist} \leq 0)$ captures the special case where the target is within the sensed freespace, and can be reached directly. It is necessary for scenarios where the target is close to the hit point $(Hit_{dist} < Step)$, thus forcing the minimal improvement $Step$ would prevent the robot from leaving the obstacle.

The condition $(Curr_{dist} - Free_{dist} \leq Best_{dist})$ guarantees that whenever the robot leaves an obstacle and hits another obstacle, the distance from its next hit point to the target $NextHit_{dist}$ will be strictly smaller than the distance from the last hit point Hit_{dist}. Furthermore it guarantees that $(Hit_{dist} - NextHit_{dist} > Step)$ for every hit point, where $Step$ is fixed (note that $Best_{dist}$ is initialized to $Hit_{dist} - Step$). The up-to-date shortest distances along the obstacle boundary $Best_{dist}$ is considered instead of $(Hit_{dist} - Step)$ in order to avoid undesirable behaviors that may occur if the the distance from the last hit point was considered (see figure 2). The parameter $Step$ bounds the number of obstacles the robot hits on its way to $N = \frac{dist(S,T)}{Step}$, where $dist(S,T)$ is the distance from the starting point S to the target T. The algorithm will terminate after at most N hit points. Given that the perimeter of every obstacle is finite the path that the robot will traverse will be finite.

Increasing the sensor range R increases $Free_{dist}$ (when no close obstacles are present) and makes it possible to leave the obstacle boundary earlier. Therefore increasing the sensor range has a direct effect on the improvement of the path (see figure 1). This behavior differs from the algorithms presented in [6] where the current obstacle would be left only if the line (S,T) is within the "radius of vision". *DistBug* uses range data directly, in contrast to modeling the local environment in [6] algorithms.

We will first present a convergence proof for the al-

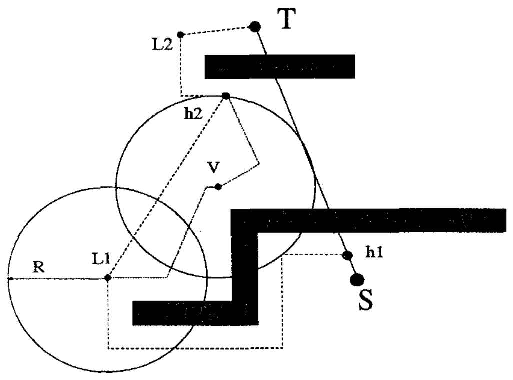

Figure 1: An example of a path planned using the distance-based leaving condition (dashed line), compared with $VisBug21$ algorithm from [5] (dotted line). In point $L1$ we have $(dist(L1,T) - Free_{dist} < Best_{dist})$ and the robot goes straight to T (note the "radius of vision" R). However $VisBug21$ algorithm would drive the robot near the obstacle boundary until the point V from which the line (S,T) is "visible".

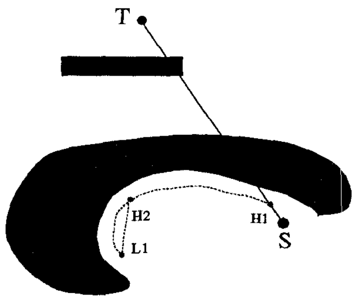

Figure 2: A problematic behavior that could occur if the leaving condition was $(Curr_{dist} - Free_{dist} \leq Hit_{dist} - Step)$: the robot leaves the obstacle in $L1$ and goes towards the target because $(dist(H1,T) - dist(H2,T) > Step)$.

gorithm, in which the parameter *Step* will be set using knowledge about the environment. When no knowledge about the environment is available setting a too big *Step* may prevent the robot from leaving obstacles, and thus from reaching the target. We will present a modified version of the leaving condition that can be used in this scenario.

Convergence proof. We assume that the minimal distance between obstacles is Min_{dist}. The value of *Step* is set accordingly: $Step \leftarrow minimum(Min_{dist}, R)$, where R is the maximal sensor range. We will show that under these conditions, the robot will reach the target if possible, and halt otherwise.

Lemma 1: *If the target is reachable from a hit point H, then the leaving conditions will enable the robot to leave the obstacle boundary and proceed towards the target.*

Proof: Consider a hit point H, where the robot reaches the obstacle O. The robot can reach any point on O's boundary in "boundary following" mode. If the target T is reachable from H, then there exist at least one point on the boundary of O from which the robot can leave the obstacle and proceed towards the target. In particular we can look at a point *Closest* on the boundary of O that is closest to the target. When the robot reaches the point *Closest* we have $(Free_{dist} > 0)$ because T is reachable from *Closest* and the straight line $[Closest, T]$ does not intersect O. $Curr_{dist}$ is the distance from *Closest* to T. If $(Curr_{dist} \leq Hit_{dist} - Step)$ then $(Best_{dist} = Curr_{dist})$ and the leaving condition $(Curr_{dist} - Free_{dist} \leq Best_{dist})$ holds. Otherwise $Best_{dist}$ has its initialization value $(Best_{dist} = Hit_{dist} - Step)$. There are two possibilities for the way from *Closest* to the target T:

1. There are no obstacles between *Closest* and T. If the target is within the sensor range R then the leaving condition $(Curr_{dist} - Free_{dist} \leq 0)$ holds. If the target is not within the sensor range then $(Free_{dist} \leftarrow R)$. The leaving condition $(Curr_{dist} - Free_{dist} \leq Hit_{dist} - Step)$ holds, because $(Curr_{dist} \leq Hit_{dist})$ and $(R \geq Step)$.
2. There is an obstacle on the straight line from *Closest* to T. It is not the current obstacle O because *Closest* is closest to the target on the boundary of O. In this case $(Free_{dist} \geq Min_{dist})$. The leaving condition $(Curr_{dist} - Free_{dist} \leq Hit_{dist} - Step)$ holds, because $(Curr_{dist} \leq Hit_{dist})$ and $(Min_{dist} \geq Step)$.

We showed that if the target is reachable then the leaving conditions will enable the robot to leave the obstacle. **qed.**

Proposition 1: *If the target is reachable from the starting point then the robot will reach the target.*

Proof: Leaving the starting point S the robot may hit at most N_0 obstacles, where $N_0 = \frac{dist(S,T)}{Step}$. The robot moves directly towards the target until reaching it or hitting an obstacle. If an obstacle is hit, Lemma 1 guarantees that the robot will leave it and proceed towards the target. From every leave point L_i the robot may hit at most N_i obstacles, where $N_i = \frac{dist(L_i,T)}{Step}$. The leaving condition $(Curr_{dist} - Free_{dist} \leq Best_{dist})$ guarantees that $N_i \leq N_{i-1} - 1$ for every $i > 0$. Therefore after k hit points $(k \leq N_0)$ there will be $N_k = 0$, and the target will be reachable directly from the leave point L_k. **qed.**

Proposition 2: *If the robot completes a loop around an obstacle then the target is unreachable.*

Proof: Follows directly from Lemma 1. If the target is reachable then the robot will leave any obstacle before completing a loop around it. Therefore if the robot completes a loop around an obstacle it means that the target is unreachable. **qed.**

Proposition 3: *The algorithm terminates.*

Proof: Given that the workspace is bounded, it is sufficient to show that the robot may hit only a bounded number of obstacles on its way. The leaving conditions enable the robot to leave obstacle boundaries only a limited number of times. The first condition $(Curr_{dist} - Free_{dist} \leq 0)$ can be used only once, because after it is used the target will be reached directly, and the algorithm will halt. The second condition $(Curr_{dist} - Free_{dist} \leq Best_{dist})$ can be used only N times, where $N = \frac{dist(S,T)}{Step}$. After the robot hits the $N + 1^{th}$ obstacle there are two possibilities: either the target can be reached directly using the first condition, or the robot will not be able to leave the obstacle. In the second case the robot will complete a loop around the obstacle and halt. **qed.**

Assuming no knowledge about the environment. In the convergence proof described above the parameter *Step* was set using some knowledge about the environment. When no knowledge about the environment is available setting a too big *Step* may prevent the robot from leaving obstacles, and thus from reaching the target (note that $Best_{dist}$ is initialized to $Hit_{dist} - Step$). To overcome this problem we add a version of the leaving condition of *Bug*2 algorithm from [7] to our distance based leaving condition, using a boolean *OR* relation. In this way the robot can always leave an obstacle. Using the modified leaving condition it is reasonable to set the parameter $Step$ for typical scenarios. For a real robot the improvement in the distance to the target between hit points will usually be larger than the robot size.

We define CROSS (line crossing) as a boolean condition that holds if the robot meets the straight line (H, T) between the last hit point H and the target T; We define the following leaving condition:

If $(Free_{dist} > 0)$ and
(((CROSS holds) and $(Curr_{dist} < Hit_{dist})$) OR
$(Curr_{dist} - Free_{dist} \leq Best_{dist})$)
then leave the obstacle.

The condition $C1 \equiv$ ((CROSS holds) and $(Curr_{dist} < Hit_{dist})$) is equivalent to the leaving condition of $Bug2$ algorithm. Our reference straight line is drawn from the last hit point H instead of the starting point S. The condition $C2 \equiv (Curr_{dist} - Free_{dist} \leq Best_{dist})$ guarantees that $(Hit_{dist} - NextHit_{dist} \geq Step)$ for every hit point. The improvement size $Step$ bounds the number of activations of this leaving condition to $N = \frac{dist(S,T)}{Step}$.

The convergence proof of the algorithm is as follows: (1) The condition $C1$ alone guarantees reaching the target in a finite path if the target is reachable (the complete proof is presented in [7]). Considering any leave point L_i as a new start, convergence is guaranteed if only $C1$ will be used after L_i. (2) The condition $C2$ can be activated at most N times during the path, thus defining at most N leave points. Claim (1) implies that after the last leave point defined by $C2$ the algorithm will converge using the leaving condition $C1$ alone.

2.2 Choosing the boundary following direction

The boundary following direction has a significant effect on the path length, especially when the perimeter of the followed obstacle is long. It is desirable that the robot would choose the boundary following direction that will result in a shorter path to the target. However, this requires global knowledge that the robot does not have. We address the problem of choosing the boundary following direction in two stages: the initial direction is chosen based on local information near the hit point. The boundary following direction can be reversed if the robot concludes that it had chosen the wrong direction.

Choosing the initial boundary following direction (clockwise/counterclockwise) based on local sensory information is similar to choosing the motion direction (left/right). A reasonable heuristic is to choose the direction where the robot can get closer to the target. If the decision is turn right, for example, the boundary following direction will be counterclockwise. It is better to make this decision before the robot reaches the obstacle, because the robot "view area" shrinks as it gets closer to the obstacle. The robot should decide while approaching an obstacle (moving in a straight line) which turning direction to use when it reaches the obstacle.

Following a purposive approach we want to choose the turning direction without building a local model of the environment. We use range readings, that are gathered as the robot approaches the obstacle, from a limited range of angles in front of the robot. The largest readings from the left and right sides are stored in the variables $Left, Right$. The variable Dir is initialized to zero, and updated in every cycle $Dir = Dir + (Left - Right)$. The value of Dir is accumulated up to a constant. The turning direction is defined by the sign of Dir. If Dir is positive then the turning direction is left (there is more freespace in the left side). Otherwise the turning direction is right. In this way information of several cycles is considered, and several cycles are usually necessary to reverse the turning direction. Big differences between range readings, indicating more global information, have a stronger effect than smaller differences that are usually more local.

The initial boundary following direction is chosen based on local information near the hit point, and may be the "wrong" direction in the sense that it will result in a longer path. The following criterion attempts to reverse a wrong decision based on higher level of information that is still local in nature (If the robot went clockwise around the obstacle, it would turn backwards and go counterclockwise, and vice versa). The criterion for reversing the boundary following direction is trigerred when the robot moves in the opposite direction to the target (more precisely, when the robot heading momentarily points to the opposite direction to the target). Changing the boundary following direction can be done only once during the following of the same obstacle, to avoid local loops. Boundary following direction should be reversed only if the robot did not go too far from the hit point. In this way the addition to the path length due to the reversed following direction will be bounded.

3 Experimental results

A Nomad200 mobile robot was used for testing. Ultrasonic, infrared and structured light sensors were used for range sensing. There are 16 ultrasonic and infrared sensors, that provide 360 degrees coverage around the robot. The structured light sensor generates a horizontal plane of light in front of the robot, and provides range readings using triangulation.

Three simulated environments with decreasing complexity were used to test the algorithms (see figure 3). "World3" is a rough description of our laboratory. "World2" was constructed from "world3" by erasing parts of the obstacles. "World1" describes a simple environment with a few separated obstacles. Nine start/target points were chosen by hand for "world2" and "world3". All the possible combinations between those points were used to generate 72 start/target pairs for testing in each environment. One hundred pairs were used for testing in "world1". The start and target location were chosen randomly: the x coordinates were fixed (left and right sides of the environment) and the y coordinates were chosen randomly within a given range.

Three versions of $DistBug$ algorithm are presented in the following experiments. Choosing the initial turning direction was used in version $DistBug1$. The criterion for reversing the boundary following direc-

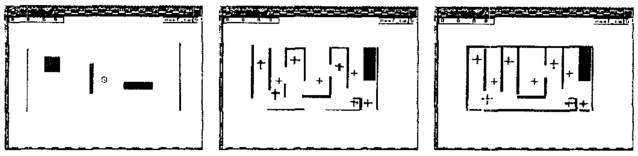

Figure 3: The simulated environments: left – "world1". The start/target points were chosen randomly from the vertical lines on both sides; middle – "world2", the nine start/target points are marked with +; right – "world3".

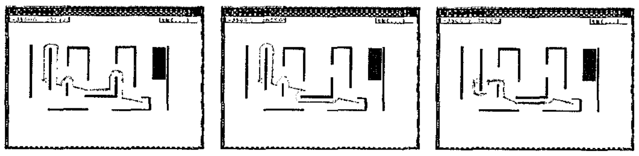

Figure 5: Simulation results in "world2" environment. Left - *Bug*2 algorithm. Middle - *DistBug*2, choosing the turning direction (path length is 0.94). Right - *DistBug*4, using the leaving condition (path length is 0.61).

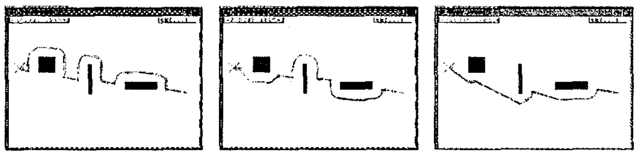

Figure 4: Simulation results in "world1" environment (target on the left side). Left - *Bug*2 algorithm. Middle - *DistBug*1, choosing the turning direction (path length is 0.84 relative to *Bug*2). Right - *DistBug*3, using the leaving condition (path length is 0.67 relative to *Bug*2).

algorithm type	world1	world2	world3
*DistBug*1	0.86	0.81	0.96
*DistBug*2	0.86	0.80	0.52
*DistBug*3	0.79	0.70	0.45

Table 1: Performance of *DistBug* algorithms. Choosing the initial turning direction was used in *DistBug*1. The criterion for reversing boundary following direction was added in *DistBug*2. The condition for leaving obstacle boundary was added in *DistBug*3. The average path length over 72 runs is presented relative to *Bug*2 performance.

tion was added to the version *DistBug*2. The condition for leaving obstacle boundary was added to the version *DistBug*3. Examples of simulated runs are presented in figures 4,5 and 6. The simulation results are summed in table 1. We used *Bug*2 algorithm from [7] as a base line for comparison.

The results show a qualitative difference between "simple" environments ("world1" and "world2") and complex ones ("world3"). In the simple scenarios many passages exist between separate obstacles, and those are "found" by the local turning direction mechanism (see first line in table 1). The obstacles are relatively simple in shape, and in most cases the robot is not driven far away from the target area. Therefore reversing the following direction has a negligible effect (see second line in table 1). In the complex scenario ("world3") there is only one connected obstacle with long perimeter and complicated shape. The local information is not sufficient to determine the globally optimal turning direction, thus choosing the direction has a small effect on performance. Choosing the correct turning direction becomes a crucial issue in complex environments, where turning to the wrong direction may increase the path length significantly. Therefore the possibility to reverse the boundary following direction improved the performance significantly (see second line in table 1). Leaving the obstacle boundary and going straight towards the target improved the path length in all the environments (see third line in table 1). It also improved significantly the average distance from obstacles, and hence produced safer paths.

The algorithm was tested in more than one hundred runs of our Nomad200 robot in three experimental scenarios: (1) using several artificial obstacles (cartoon boxes). The obstacles were usually grouped into 2-3 simple shaped obstacles (similar to "world1" setting). (2) working in unchanged laboratory environment. The obstacles were walls, tables, chairs, cupboards etc. (see figure 7) (3) adding artificial obstacles to the laboratory environment. The algorithm was successful in most cases, driving the robot to the target location. However several problems were noticed. The range sensors (sonar, infrared and structured light) have low reliability in the laboratory environment. As a result the robot might take wrong decisions and the convergence of the algorithm is not guaranteed (for example - leaving an obstacle boundary assuming that there are no obstacles in the direction to the target). Moving obstacles, i.e. people, disturb the robot motion if they are detected during boundary following behavior. The robot tries to follow these obstacle, but can not find them as they go away. The problem is partially solved by a procedure that looks for "lost obstacles", and drives the robot back to the previously followed obstacle after such a disturbance. However a better analysis that will distinguish between moving and static obstacles is necessary in order to react differently to stationary and moving obstacles. Reaching a person while moving in straight line did not cause a problem in most cases: the robot tried to bypass the obstacle, and as the obstacle disappeared it concluded that it can go straight to the target again.

4 Summary and conclusions

We presented a sensory based algorithm *DistBug* that is guaranteed to reach the target if possible, and report if the target is unreachable. The algorithm is reactive in the sense that it relies on range data to make local decisions, and does not create a world model. The algorithm consists of two behav-

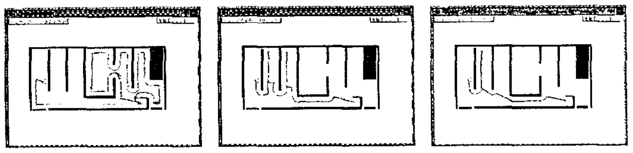

Figure 6: Simulation results in "world3" environment. Left - *Bug*2 algorithm. Middle - *DistBug*2, choosing the turning direction + reversing boundary following direction (path length is 0.54). Right - *DistBug*3, using the leaving condition (path length is 0.36).

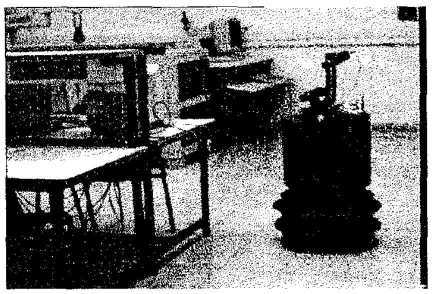

Figure 7: The Nomad200 robot in a typical laboratory environment.

iors (modes of motion): straight motion between obstacles and obstacle boundary following. Simulation results as well as experiments with a real robot were presented.

The condition for leaving obstacle boundary is based on the free range in the direction to the target. This condition allows the robot to leave the obstacle as soon as the local conditions guarantee global convergence. Range data is utilized for choosing the initial boundary following direction when the robot approaches an obstacle. A criterion for reversing the boundary following direction when it seems to be the wrong direction is also introduced. As a direct result of these local decisions a significant improvement in the performance was achieved.

The simulation results showed a qualitative difference between simple environments and complex ones. The environment simplicity is related to the obstacles shape, that affects the usefulness of local decisions, and to the length of the obstacles perimeter, that affects the penalty for choosing the wrong turning direction. Choosing the initial turning direction was useful in simple environment. Reversing the boundary following direction was useful in complex environments. Leaving the obstacle as soon as the local conditions guarantee global convergence improved the performance in all the environments. In our simple environment the average path length over 100 runs was reduced to 0.79 (relative to *Bug*2 performance), the average traveling time was reduce to 0.67, and the minimal distance from obstacles (safety measure) was increased to 1.6. In the complex environment the average path length was reduced to 0.45.

Acknowledgments

We would like to thank Michael Heymann and Freddy Bruckstein for very helpful discussions. We want to express special thanks to Elon Rimon for his encouragement and helpful comments.

References

[1] R. C. Arkin. Motor schema based navigation for a mobile robot: an approach for programming by behavior. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–271, 1987.

[2] James L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 674–680, 1989.

[3] James L. Crowley and Yves Demazeau. Principles and techniques for sensor data fusion. *Signal Processing*, 32:5–27, 1993.

[4] S. G. Goodridge and R. C. Luo. Fuzzy behavior fusion for reactive control of an autonomous mobile robot: Marge. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1622–1627, 1994.

[5] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 500–505, 1985.

[6] V. J. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE transactions on Systems, Man, and Cybernetics*, 20(5):1058–1068, 1990.

[7] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape. *Algoritmica*, 2:403–430, 1987.

[8] V. J. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 111–116, 1994.

[9] H. P. Moravec and A. Elfes. High resolution maps from angle sonar. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 116–121, 1985.

[10] H. Noborio and T. Yoshioka. An on-line and deadlock-free path-planning algorithm based on world topology. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 1425–1430, 1993.

[11] P. Reignier. Molusc: an incremental approach of fuzzy learning. In *Proceedings of the International symposium on Intelligent Robotics Systems*, pages 178–186, 1994.