Sensory-Based Motion Planning with Global Proofs

Ishay Kamon, *Student Member, IEEE*, and Ehud Rivlin, *Member, IEEE*

***Abstract*— We present *DistBug*, a new navigation algorithm for mobile robots which exploits range data. The algorithm belongs to the *Bug* family, which combines local planning with global information that guarantees convergence. Most *Bug*-type algorithms use contact sensors and consist of two reactive modes of motion: moving toward the target between obstacles and following obstacle boundaries. *DistBug* uses range data in a new "leaving condition" which allows the robot to abandon obstacle boundaries as soon as global convergence is guaranteed, based on the free range in the direction of the target. The leaving condition is tested directly on the sensor readings, thus making the algorithm simple to implement. To further improve performance, local information is utilized for choosing the boundary following direction, and a search manager is introduced for bounding the search area. The simulation results indicate a significant advantage of *DistBug* relative to the classical *Bug2* algorithm. The algorithm was implemented and tested on a real robot, demonstrating the usefulness and applicability of our approach.**

***Index Terms*—Mobile robots, sensor-based navigation.**

I. Introduction

AUTONOMOUS navigation of indoor mobile robots has received considerable attention in recent years. Work in this area was motivated by applications such as office cleaning, cargo delivery, etc. In realistic settings, the robot cannot base its motion planning on complete *a priori* knowledge of the environment. The robot must rather use its sensors to perceive the environment and plan accordingly. The two main sensor-based motion planning approaches use either global planning or local planning. Let us briefly describe these approaches and point out their limitations.

In the global sensor-based planning approach, the mobile robot builds a global world model based on sensory information and uses it for path planning [6], [20], [21]. This approach guarantees global convergence to the target. However, the construction and maintenance of a global model based on sensory information imposes a heavy computational burden on the robot. Moreover, the reliance on a global model for navigation requires frequent localization of the robot relative to the model, a process which is difficult to attain due to the inherent uncertainties of practical sensors [5], [10], [16]. Recent works use the global approach to achieve sensor-based navigation of general robots [4], [17].

In contrast, local path-planners use local sensory information in a largely reactive fashion. They are much simpler to implement than global planners, since they typically map the sensor readings directly to actions. Various examples include potential-field methods [1], [8], fuzzy logic approaches [7], [15], and specialized approaches [2], [3]. As with any local search method, local path-planners do not guarantee global convergence to the target since they may get trapped in local minima.

Thus, the global approaches are difficult to implement, while the local ones lack a global convergence guarantee. This paper focuses on a midway approach, called the *Bug* approach, which was originated by Lumelsky and Stepanov [13], and subsequently studied in [11], [14], and [18]. This approach combines local planning with a globally convergent criterion as follows. Initially, the robot moves directly toward the target. When the robot hits an obstacle it starts to follow the obstacle boundary. The robot leaves the obstacle boundary and resumes motion toward the target only when a *leaving condition*, which monitors a globally convergent criterion, holds. The *Bug* approach reduces the reliance on a global model to the essential minimum of loop detection while augmenting the purely reactive navigation decisions with a globally convergent criterion. This approach thus minimizes the computational burden on the planner while still ensuring global convergence to the target. However, the *Bug* algorithms mainly use contact sensors. Range data was incorporated only at a later stage in an algorithm termed *VisBug* [12], which calculates shortcuts relative to the path generated by the *Bug2* algorithm from [13], or to the line [*Start,Target*] (Fig. 1).

This paper presents a new *Bug* algorithm, termed *DistBug*, which specifically exploits range data. Our main contribution is a new leaving condition which allows the robot to abandon obstacle boundaries as soon as global convergence is guaranteed, based on the free range in the direction of the target. To further improve performance, local information is utilized for choosing the boundary following direction, and a search manager is introduced for bounding the search area. As a direct result of these extensions, a significant improvement in the performance has been achieved.

The rest of this paper is organized as follows: in Section II we present the *DistBug* algorithm, show that it is globally convergent, and provide an upper bound for its path length. In Section III we present the experimental results. Finally, the conclusions are presented in Section IV.

II. The *DistBug* Algorithm

The *DistBug* algorithm navigates a point robot in a planar unknown environment populated by stationary obstacles with arbitrary shape. The robot is equipped with a range sensor with maximal detection range R. The *DistBug* algorithm

Manuscript received March 2, 1995; revised November 1, 1995 and December 17, 1996. This paper was recommended for publication by Associate Editor M. Hebert and Editor A. J. Koivo upon evaluation of the reviewer's comments.

The authors are with the Center for Intelligent Systems, Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel.

Publisher Item Identifier S 1042-296X(97)07140-1.

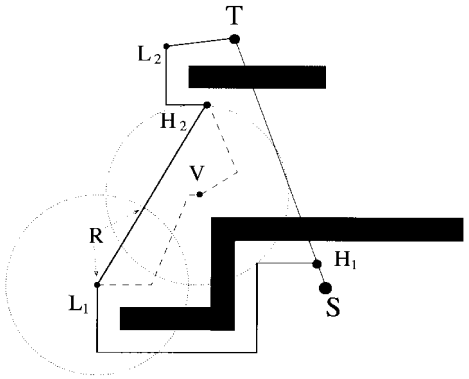


Fig. 1. Comparing paths planned by the *DistBug* algorithm (solid line) and the *VisBug* algorithm (dashed line). Using *DistBug*, the leaving condition holds at L_1. However, *VisBug* would follow the obstacle boundary until the point V, from which the line $[S,T]$ is visible.

uses two basic modes of motion: *motion toward the target* and *obstacle-boundary following*. Initially, the robot moves directly toward the target until it hits an obstacle. It then switches to the boundary following mode and moves along the obstacle boundary. During boundary following, the robot records the minimal distance to the target $d_{\min}(T)$ achieved since the last hit point. It also senses the distance in freespace F from the current location X to the nearest obstacle in the direction of the target. If no obstacles are detected F is set to R. The robot leaves the obstacle boundary when either the target becomes visible or $d(X,T) - F \leq d_{\min}(T) - Step$ holds, where $d(X,T)$ is the distance from X to the target T and *Step* is a predefined constant. We now describe the *DistBug* algorithm in detail.

1) Move toward the target until one of the following events occurs.
   a) The target is **reached**. Stop.
   b) An obstacle is reached. Go to Step 2.
2) Follow the obstacle boundary while recording the minimal distance to T $d_{\min}(T)$ and sensing the distance in freespace F, until one of the following events occurs.
   a) The target is visible: $d(X,T) - F \leq 0$. Go to Step 1.
   b) The range-based leaving condition holds: $d(X,T) - F \leq d_{\min}(T) - Step$. Go to Step 1.
   c) The robot completed a loop around the obstacle. The target is **unreachable**. Stop.

Next we elaborate on the leaving condition. Motivated by the fact that moving along a straight path is faster and safer than boundary following, the leaving condition is designed to abandon the boundary as soon as convergence is guaranteed. The leaving condition holds when one of the following terms is satisfied: either $(\boldsymbol{d(X,T) - F \leq 0})$ or $(\boldsymbol{d(X,T) - F \leq d_{\min}(T) - Step})$. The first term, $d(X,T) - F \leq 0$, is triggered when the target becomes visible and can be reached directly. The second term, $d(X,T) - F \leq d_{\min}(T) - Step$, guarantees that the distance to the target decreases by at least *Step* between successive hit points, $d(H_i,T) - d(H_{i+1},T) \geq Step$. The leaving condition is based on the minimal distance achieved along the followed boundary, $d_{\min}(T)$, and not on $d(H_i,T)$, to prevent a scenario in which the next hit point is located on a part of the boundary which was already traversed. The size of the parameter *Step* is discussed in Section II-A. Note that when the leaving condition holds it is physically possible to move directly toward the target, since $F > 0$ necessarily holds.

The *DistBug* algorithm has several practical advantages over existing *Bug* algorithms. It is simple to implement because it uses range data directly, in contrast to modeling the local environment in *VisBug*. Compared to the *Bug2* and *VisBug* algorithms from [12] and [13], the generated paths are closer to the optimal ones since the leaving condition is not based on the line $[S,T]$, and increasing the sensor range allows to leave obstacle boundaries earlier (Fig. 1). To guarantee convergence to the target, the *DistBug* algorithm needs small amount of global information. Global positioning is necessary only during boundary following, for updating $d_{\min}(T)$ and for determining that the robot completed a loop around an obstacle. (Similar to the assumptions underlying purely local planners, we assume here that the direction from the current location to the target is known, and that the robot can determine when the target is reached.) The leaving condition is robust with respect to noise in $d_{\min}(T)$, because the parameter *Step* forces a significant improvement in the distance to the target. Choosing *Step* larger than the expected position error guarantees convergence in the presence of noise.[1] Note that *DistBug* is purely reactive when the target is visually tracked, because in this case $d_{\min}(T)$ can be extracted directly from the visual information.

A. Algorithm Analysis

First, we analyze the convergence of the *DistBug* algorithm under the assumption that the minimal distance between obstacles is M. In Section II-B we will present a modified version of the leaving condition, which can be used when no *a priori* knowledge about the environment is available. The value of the parameter *Step* is set to $minimum(M,R)$, where R is the maximal detection range. We also assume that the perimeter of each obstacle is finite. Next, we prove that *DistBug* is complete and give an upper bound on its performance.

Lemma 2.1: If the target T is reachable from a hit point H, the leaving condition will cause the robot to leave the obstacle after a finite-length path.

Proof: Starting from H, the robot uses the boundary following behavior to move along the obstacle boundary. It will eventually reach a point C which is closest to T along the boundary. At this point $d(X,T) = d_{\min}(T) = d(C,T)$. Since T is reachable, it must be possible to move from C directly toward T, hence $F > 0$. We next show that the leaving condition holds at C, considering the following three cases. If the target is visible then the term $d(X,T) - F \leq 0$ holds. If the target is not visible and no obstacle is detected in the direction of the target then $F = R$. In this case the term $d(X,T) - F \leq d_{\min}(T) - Step$ holds because $Step \leq R$. If an obstacle is

[1] As we discuss in Section II-A, *Step* should be smaller than the minimal distance between obstacles. Otherwise, a modified leaving condition, which we present in Section II-B, should be used.

detected in the direction of the target, $F \geq M$ holds because the detected obstacle must be different from the currently followed one. In this case $d(X,T) - F \leq d_{\min}(T) - Step$ holds because $Step \leq M$. □

Theorem 1: DistBug always ***terminates*** after following a finite-length path.

Proof: The path generated by the algorithm consists of motion toward the target segments and boundary following segments. The path length of each motion toward the target segment is finite because it is a straight line pointing toward the target. The path length of each boundary following segment is bounded by the perimeter of the followed obstacle. To prove that the algorithm terminates after following a finite-length path, we show that there is a finite number of motion segments. It is sufficient to show that the leaving condition enables the robot to leave obstacles only a finite number of times. The first term, $d(X,T) - F \leq 0$, can be used at most once, because the robot can reach the target directly after this condition holds. The second term, $d(X,T) - F \leq d_{\min}(T) - Step$, can be used at most N times, where $N = ||S - T||/Step$. After the robot hits the $N+1$th obstacle it has two possibilities: either the target will be reached directly using the first term, 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. Hence there is a finite number of motion segments, and the algorithm terminates after following a finite-length path. □

Theorem 2: DistBug ***finds the target*** if it is reachable from the start point.

Proof: Every motion toward the target segment terminates either at the target or at a hit point H. If T is reachable from H, Lemma 2.1 guarantees that every boundary following segment terminates at a leave point. Since the number of boundary following segments is finite and every such segment is followed by a motion toward the target segment, there is a *last* motion toward the target segment. This last segment terminates at the target.

Proposition 2.2: An upper bound $L_{\max}$ on the path length that *DistBug* generates is

$$L_{\max} = ||S - T|| + N' \times (P + R)$$

where $N' = (||S - T||/Step) + 1, P$ is the maximal obstacle perimeter from the obstacles intersecting the disc of radius $||S - T||$ centered at T, and R is the maximal sensor range.

Proof: To bound the path length of motion toward the target segments, we introduce new notations. Every *leave point* L_i is associated with a *reach point* G_i, to which the distance in freespace F is measured along the line $[L_i, T]$, so that $d(G_i,T) = d(L_i,T) - F$. A reach point G_i is located either in freespace $(F = R)$ or on an obstacle boundary $(F \leq R)$. The leaving condition guarantees that $d(G_i,T) < d(L_i,T) < d(H_i,T)$, and path construction guarantees that $d(G_i,T) \geq d(H_{i+1},T)$ because either G_i is the next hit point or the robot proceeds from G_i toward the target. We denote the starting point S as G_0 and the target T as H_k for some $k \leq N'$. The accumulated sum of $[G_i, H_{i+1}]$ segments is bounded by $||S - T||$ since the start point of each segment is closer to the target than the endpoint of the previous one, and these segments all point toward T. The accumulated sum of $[L_i, G_i]$ segments is bounded by $N' \times R$, because the leaving condition may hold at most $||S - T||/Step$ times, and the length of each segment is bounded by the detection range R. Thus $||S - T|| + N' \times R$ bounds the path length of motion toward the target segments.

The path length of each boundary following segment is bounded by the perimeter of the followed obstacle. The robot may hit at most N' obstacles, because the leaving condition may hold at most $||S - T||/Step$ times. All the hit points are contained in the disc of radius $||S - T||$ centered at T, because the distance to the target decreases along motion toward the target segments. Thus the term $N' \times P$ bounds the path length of boundary following segments.

This upper bound is comparable with the upper bound for *Bug2* from [13], because the robot may hit each obstacle k times, where k is not fixed. We may also ask what is the lower bound. In [18], Sankaranarayanan and Vidyasagar show that the worst-case lower bound on the path length of *Bug*-type algorithms is $||S - T|| + 2\Sigma_i \Pi_i$, where Π_i is the perimeter of the ith obstacle which intersects the disc of radius $||S - T||$ centered at T. They also propose an algorithm which uses a contact sensor and achieves this bound as its upper bound. But their algorithm is not truly reactive, as it maintains a *global* data structure of all the hit and leave points. In contrast, *DistBug* stores only the last hit point. Moreover, the same global data can be incorporated into *DistBug*, with a similar effect of a better bound on the path length.

B. Assuming No Knowledge about the Environment

When no *a priori* knowledge about the environment is available, setting too big a *Step* may prevent the robot from leaving obstacles, and thus from reaching the target. To overcome this problem we present a modified version of the leaving condition. We add a version of the leaving condition from *Bug2* algorithm to our range-based leaving condition, using a boolean OR relation. In this way the robot can always leave an obstacle. 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 subconditions: $\boldsymbol{C1} \equiv \textbf{CROSS holds}$ and $\boldsymbol{d(X,T) < d(H,T), C2 \equiv d(X,T) - F \leq d_{\min}(T) - Step}$. The modified leaving condition is **C1 OR C2**.

A sketch of the convergence proof for the modified leaving condition now follows.

The condition $C1$ alone guarantees reaching the target in a finite length path if the target is reachable (the complete proof is presented in [13]). Considering any leave point L_i as a new start, convergence is guaranteed if $C1$ alone will be used after L_i. The condition $C2$ can be activated at most $N = ||S - T||/Step$ times along the path, thus defining at most N leave points. After the last leave point defined by $C2$, the algorithm will converge using the subcondition $C1$ alone.

C. Using Local Information and Search Management

In the following we present several extensions to the algorithm, which proved to be very effective in our experiments. First we describe a method for choosing the initial boundary

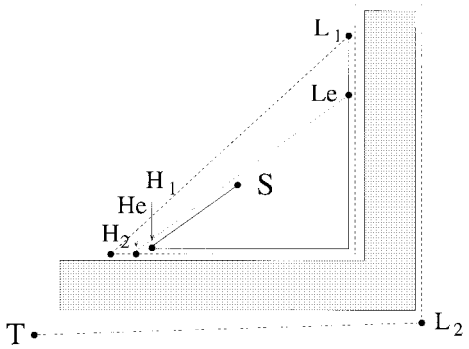


Fig. 2. Moving from S to T, the robot hits the obstacle in H_1 and follows the boundary until L_1. It then moves to H_2 and follows the boundary until L_2, from which it moves toward T.

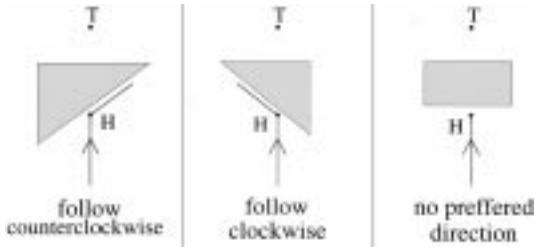


Fig. 3. Choosing the initial boundary following direction based on orientation at the hit point.

following direction based on local information, and explain why it is appropriate to use this method together with the range-based leaving condition. We then describe a local criterion for reversing the boundary following direction and a search manager for bounding the search area.

Partial use of local information may cause undesirable behavior. For a class of typical scenarios, using the proposed leaving condition while keeping a fixed boundary following direction causes the robot to traverse several times the same part of the boundary, as shown in Fig. 2. In this example we assume infinite sensor range and the boundary following direction is clockwise. Moving from S to T, the robot hits the obstacle in H_1 and follows its boundary until L_1 where the leaving condition holds, because $d(H_2,T) < d(H_1,T) - Step$. The robot then moves from L_1 to H_2 and follows the boundary until L_2, from which it moves directly toward T. Note that the value of *Step* effects the path length in this scenario. A smaller *Step* would cause the robot to perform more cycles, defining hit points between H_1 and H_2 (see for example the leave point L_e and its corresponding hit point H_e). Using range data to choose the boundary following direction would significantly reduce the path length in this scenario. In the example presented above, the local information $d(H_2,T) < d(H_1,T) - Step$, which triggers the leaving condition at L_1, is available to the robot before reaching H_1. Taking this information into account, the robot would choose the counterclockwise direction for boundary following from H_1. Based on this observation and considering $Step \mapsto 0$, we choose the initial boundary following direction based on the boundary orientation at the hit point. The robot turns to the direction which takes it closer to the target (Fig. 3).

Local information can also be used during boundary following. We reverse the following direction when the current heading drives the robot away from the target, thus indicating that some part of the followed obstacle does not block the way from the robot to the target. (Similar considerations, based on a local map, are used in [19]). In our experiments, the local reversing criterion is triggered when the angle between the robot heading and the direction toward the target exceeds 135°. The boundary following direction is reversed at most once after each hit point, to avoid oscillations.

The problem of path planning with incomplete knowledge can be viewed as a search problem [11]. To facilitate the search we present a method for bounding the search area by "virtual obstacles," which takes inspiration from the iterative deepening approach introduced in [9]. The robot first performs an exhaustive search within the bounded area. If the target is not found then the search area is enlarged. We use circles, centered at the target, as the virtual obstacles. A new virtual obstacle is defined whenever the robot hits an obstacle. When the robot first touches the virtual obstacle it reverses the boundary following direction. If the robot touches the virtual obstacle for the second time, it concludes that the target is unreachable within the current virtual obstacle, and consequently enlarges the search area. Note that the virtual obstacles are used as criteria for reversing the boundary following direction. Compared to the local criterion for reversing boundary following direction, the search manager is more robust with respect to local disturbances.

The extensions described above do not ruin the convergence of the *DistBug* algorithm. Choosing the initial boundary following direction has no influence on convergence. Using the local criterion for reversing the following direction, the direction can be reversed at most once after each hit point. Thus the robot can traverse the boundary at most twice after each hit point, and convergence is not disturbed. The search manager does not ruin convergence, because the search area is enlarged whenever the robot concludes that it is blocked within a virtual obstacle. Consequently, if a path to the target exists, it will eventually be contained within the search area. On the other hand, if the target is unreachable, the entire boundary segment which blocks the way to the target will eventually be contained within the search area. Therefore the robot will complete a loop around that boundary segment, conclude that the target is unreachable, and halt.

III. Experimental Results

The experimental study of the *DistBug* algorithm consists of simulations and experiments on a mobile robot. Simulations were performed to study the effect of the range-based leaving condition and the various extensions on the resulting paths. The simulations compared the *DistBug* algorithm with the classical *Bug2* algorithm, showing that *DistBug* generates shorter and safer paths. The algorithm was also implemented on a Nomad200 robot, demonstrating the usefulness and applicability of our approach.

The performance of the *DistBug* algorithm was evaluated considering the average path length, which will be discussed below, and path safety, which will be discussed later. The

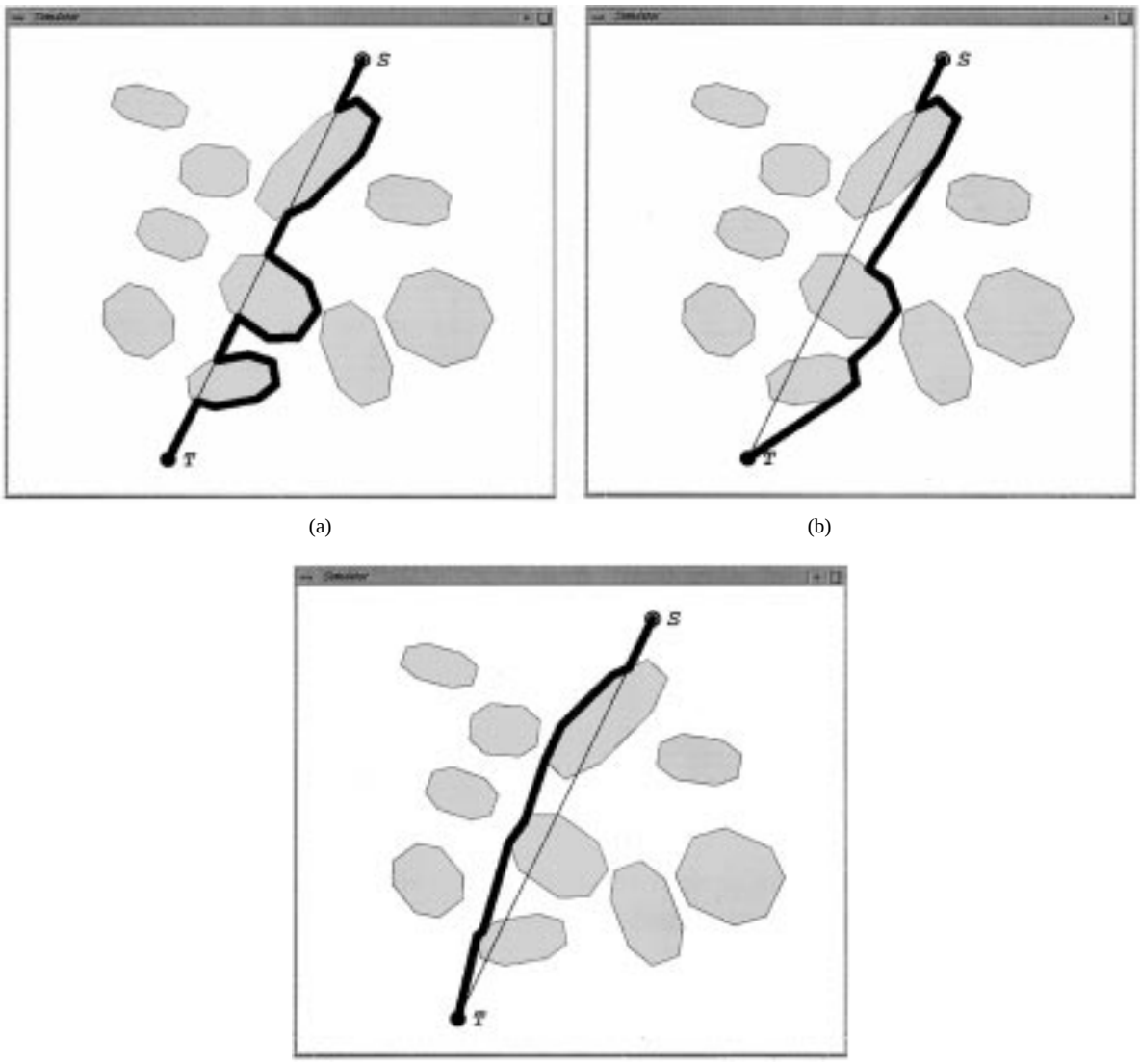


Fig. 4. Simulation results in "*world1*" environment. (a) *Bug2* algorithm. (b) *DistBug* algorithm (path length is 0.75 relative to *Bug2*). (c) *DistBug* + choosing the boundary following direction (path length is 0.62).

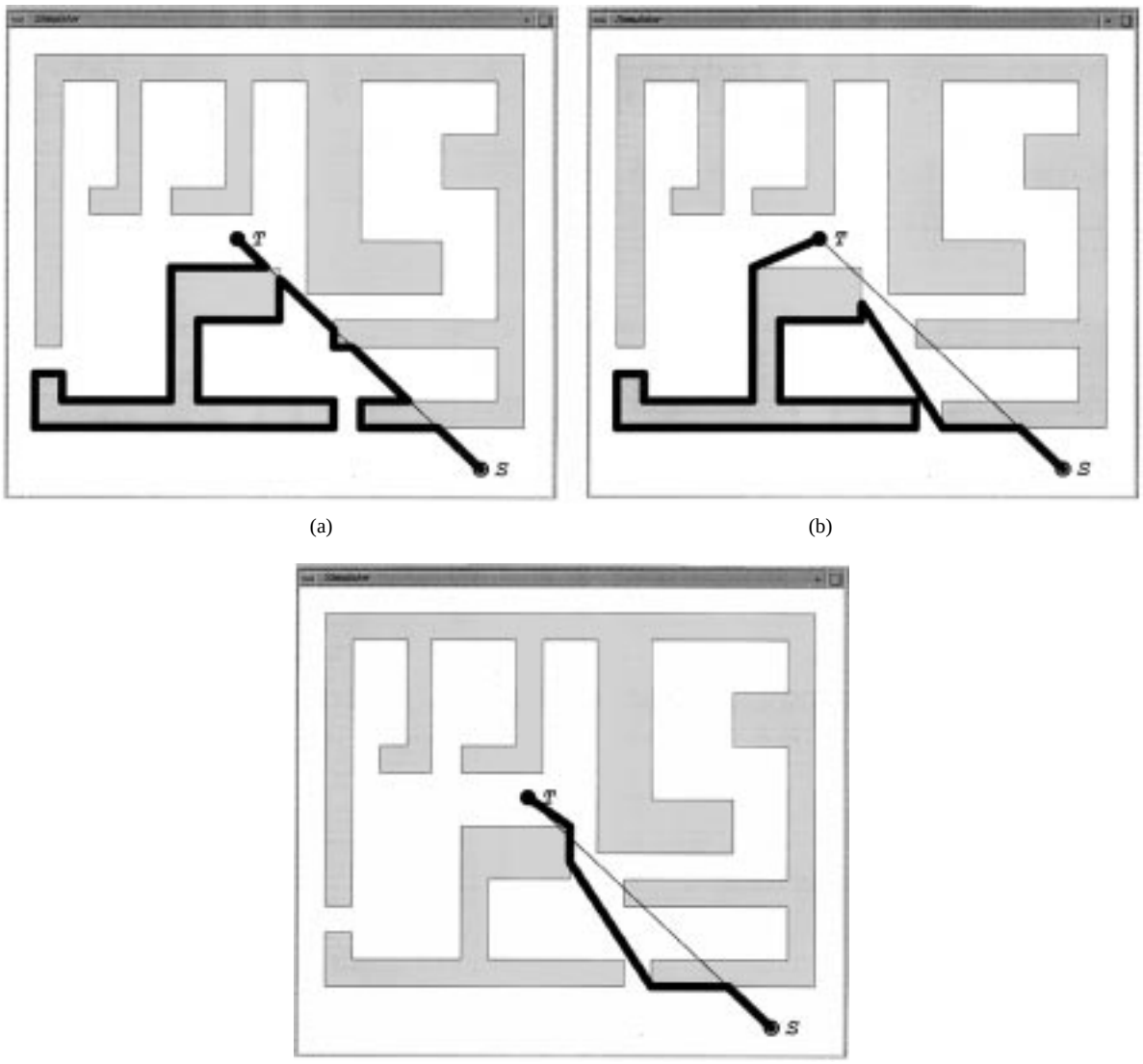
algorithm was tested in two simulated environments. The simple environment "*world1*" consisted of convex nonintersecting obstacles (Fig. 4), while the complex environment "*world2*" consisted of concave obstacles with an "office-like" shape (Fig. 5). Unlimited sensor range was assumed in all the experiments, and the parameter *Step* was chosen as the minimal distance between obstacles.

The results of path length comparison between plain versions of *DistBug* and *Bug2* algorithms are presented in the first line of Table I. The table contains the average path length over 100 runs in each environment, with randomly chosen start/target points, relative to the path length generated by *Bug2*. One can see that the *DistBug* algorithm generates shorter paths in both environments. A more significant improvement in the path length was achieved when local decisions and a search manager were added to the plain algorithm. We added our modifications one at a time to both *DistBug* and *Bug2*, and tested 100 runs in each environment. The results are summed up in Table I.

TABLE I
AVERAGE PATH LENGTH OF THE *DistBug* ALGORITHM

	world1		world2	
Type	*Bug2*	*DistBug*	*Bug2*	*DistBug*
P	1.00	0.92	1.00	0.89
P+D	0.84	0.78	0.94	0.71
P+D+Rv	0.84	0.78	0.61	0.42
P+D+SM	0.84	0.78	0.49	0.41

First, we added the method for choosing the initial boundary following direction (denoted D in Table I). In the complex environment, "*world2*," the combination of this method and the range-based leaving condition generates paths which are significantly shorter than those generated by the *Bug2* algorithm with the same modification. Next we added the local criterion for reversing the boundary following direction (denoted Rv in Table I). The reversing criterion caused a



(a) (b)

(c)

Fig. 5. Simulation results in "*world2*" environment. (a) *Bug2* algorithm. (b) *DistBug* algorithm (path length is 0.87 relative to *Bug2*). (c) *DistBug* + choosing the boundary following direction (path length is 0.24).

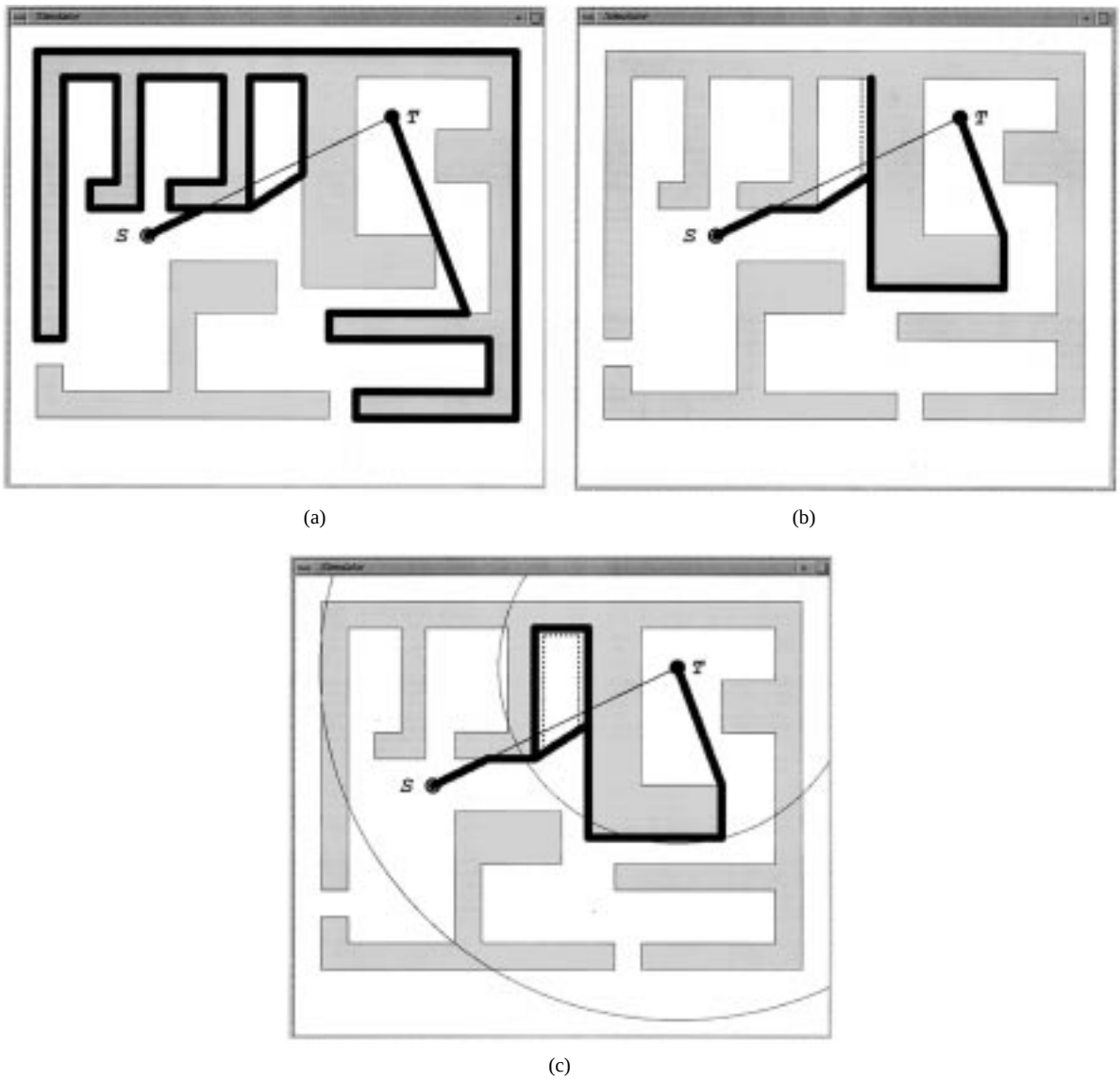
significant improvement in "*world2*," but did not affect the results in "*world1*," since in most cases the convex obstacles did not drive the robot away from the target. The search manager (denoted SM in Table I) was tested with the method for choosing the initial boundary following direction. The search manager generated results similar to the local reversing criterion (Fig. 6). Encouraged by the significant improvement in performance of the *Bug2* algorithm with the search manager, we believe that this mechanism can be successfully incorporated into other path-planning algorithms.

Path safety is an important property, which should be considered while evaluating path quality. We designed the following measure for path safety. The minimal distance from the robot to the surrounding obstacles was measured from every location along the path. The path safety for the entire path was defined as the average of this local safety distance. The bigger the average distance was—the safer was the path. Table II presents safety measures. The results show that the range-based leaving condition and the various extensions enlarged the average distance from obstacles, and hence produced safer paths.

TABLE II
SAFETY MEASURE OF THE $DistBut$ ALGORITHM

	world1		world2	
Type	*Bug2*	*DistBug*	*Bug2*	*DistBug*
P	1.00	1.16	1.00	1.24
P+D	1.16	1.29	1.12	1.43
P+D+Rv	1.16	1.29	1.19	1.52
P+D+SM	1.16	1.29	1.20	1.50

To conclude, the results show that the *DistBug* algorithm generates paths which are significantly shorter and safer than *Bug2* paths when local information is used to choose the boundary following direction. Moreover, the *DistBug* algorithm regularly generates shorter paths: using local informa-



(a) (b)

(c)

Fig. 6. Reversing the boundary following direction. (a) *DistBug* + choosing the initial boundary following direction. In the second hit point the robot turned left, the "wrong" direction. (b) the path planned with the local reversing criterion. Note that the following direction was reversed in the upper corner, and the robot traversed a part of the boundary twice (path length is 0.22 relative to the original path). (c) the path planned with the search manager. The robot reversed the following direction when it touched the virtual obstacle for the first time (path length is 0.34).

tion, it performed better than *Bug2* in 90% from the cases in "*world1*," and in 84% from the cases in "*world2*."

A. Experiments in a Real-World Scenario

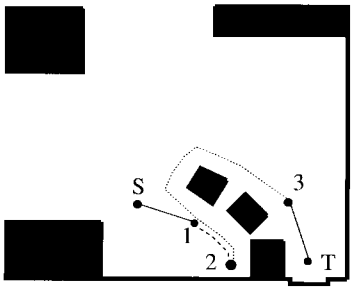
The *DistBug* algorithm was implemented and tested in more than 100 runs of our Nomad200 robot, demonstrating the simplicity and robustness of our approach. No model of the world was created, and the decisions were based directly on range data. The algorithm was successful in almost all the cases, driving the robot to the target location. However, several implementation problems were noticed. Most of the problems originated from the low reliability of the range sensors (sonar, infrared, and structured light) in the unstructured laboratory environment.

One of the experimental settings is presented in detail below. Several boxes created an oblique "wall" between the starting location S and the target location T (Fig. 7). The robot reached the obstacles in point 1 and turned right. The robot then followed the obstacles until reaching the wall in point 2, where it turned away from the target. At that point the boundary following direction was reversed. The robot turned around and followed the obstacles boundary until point 3, in which the leaving condition was satisfied. From point 3 the robot moves straight to the target T.

IV. Summary and Conclusions

We have presented *DistBug*, a new navigation algorithm for mobile robots which exploits range data. The algorithm belongs to the *Bug* family, which combines local planning with global information that guarantees convergence. The *DistBug* algorithm uses range-data in a new "leaving condition" which allows the robot to abandon obstacle boundaries as soon as global convergence is guaranteed, based on the free range in the direction of the target. We have proved the completeness

(a)



(b)

Fig. 7. (a) The Nomad200 robot in the starting point of the experiment described below. (b) A scheme of the generated path.

of the *DistBug* algorithm and derived an upper bound for the path length generated by it.

The *DistBug* algorithm uses local information in a greedy way, and thus performs well in typical environments. The leaving condition allows the robot to abandon obstacles regardless of the straight line [*Start, Target*] which was the backbone of the *Bug2* and *VisBug* algorithms from [13] and [12]. Moreover increasing the sensor range allows the robot to leave obstacle boundaries earlier. To further improve performance, we tested several extensions: local information was utilized for choosing the boundary following direction and a search manager was introduced for bounding the search area. The simulation results indicate a significant advantage of the *DistBug* algorithm relative to the algorithm *Bug2* from [13]. Moreover, the results show that the advantage of the range-based leaving condition becomes more apparent when local information is used for choosing the boundary following direction, as we have explained in Section II-C.

The *DistBug* algorithm has several practical advantages over existing *Bug* algorithms. It is simple to implement because the leaving condition is tested directly on the range readings. Global positioning is necessary only during boundary following, for updating $d_{\min}(T)$ and for determining that the robot completed a loop around an obstacle. The leaving condition is robust with respect to noise in the minimal distance to the target, $d_{\min}(T)$. The algorithm was implemented and tested on a real robot, demonstrating the usefulness and applicability of our approach.

ACKNOWLEDGMENT

The authors would like to thank M. Heymann and A. Bruckstein for very helpful discussions. The authors want to express special thanks to E. 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 *IEEE Conf. Robot. Automat.*, 1987, pp. 264–271.
[2] R. Bauer, W. Feiten, and G. Lawitzky, "Steer angle field: An approach to robust maneuvering in cluttered, unknown environments," *Robot. Autonomous Syst.*, vol. 12, pp. 209–212, 1994.
[3] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in *IEEE Conf. Robot. Automat.*, 1990, pp. 572–577.
[4] H. Choset and J. W. Burdick, "Sensor-based planning, Part II: Incremental construction of the generalized Voronoi graph," in *IEEE Conf. Robot. Automat.*, Nagoya, Japan, May 1995.
[5] J. L. Crowley and Y. Demazeau, "Principles and techniques for sensor data fusion," *Signal Processing*, vol. 32, pp. 5–27, 1993.
[6] G. Foux, M. Heymann, and A. Bruckstein, "Two-dimensional robot navigation among unknown stationary polygonal obstacles," *IEEE Trans. Robot. Automat.*, vol. 9, pp. 96–102, 1993.
[7] S. G. Goodridge and R. C. Luo, "Fuzzy behavior fusion for reactive control of an autonomous mobile robot: Marge," in *IEEE Conf. Robot. Automat.*, 1994, pp. 1622–1627.
[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE Conf. Robot. Automat.*, 1985, pp. 500–505.
[9] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intell.*, vol. 27, pp. 97–109, 1985.
[10] J. J. Leonard and H. F. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robots Navigation*. Boston, MA: Kluwer, 1992.
[11] V. J. Lumelsky, "A comparative study on the path length performance of maze-searching and robot motion planning algorithms," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 57–66, 1991.
[12] V. J. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 1058–1068, 1990.
[13] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403–430, 1987.
[14] H. Noborio and T. Yoshioka, "An on-line and deadlock-free path-planning algorithm based on world topology," in *IEEE/RSJ Conf. Intell. Robots Syst., IROS*, 1993, pp. 1425–1430.
[15] P. Reignier, "Molusc: An incremental approach of fuzzy learning," in *Int. Symp. Intell. Robot. Syst.*, 1994, pp. 178–186.
[16] W. D. Rencken, "Concurrent localization and map building for mobile robots using ultrasonic sensors," in *IEEE/RSJ Conf. Intell. Robots Syst.*, 1993, pp. 2192–2197.
[17] E. Rimon, "Construction of c-space roadmaps from local sensory data: What should the sensors look for?" *Algorithmica*, vol. 17(4), 1997, pp.357–379.
[18] A. Sankaranarayanan and M. Vidyasagar, "Path planning for moving a point object amidst unknown obstacles in a plane: The universal lower bound on worst case path lengths and a classification of algorithms," in *IEEE Conf. Robot. Automat.*, 1991, pp. 1734–1941.
[19] M. G. Slack, "Fixed computation real-time sonar fusion for local navigation," in *IEEE Conf. Robot. Automat.*, 1993, pp. 123–129.

[20] A. Stentz, "Optimal and efficient path planning for partially known environments," in *IEEE Conf. Robot. Automat.*, 1994, pp. 3310–3317.
[21] A. Zelinsky, "Using path transforms to guide the search for findpath in 2-D," *Int. J. Robot. Res.*, vol. 13, 1994, pp. 315–325.

Ishay Kamon (S'96) received the B.A. degree in computer science from Technion, Haifa, Israel, in 1989. He received the M.Sc. degree in applied mathematics and computer science from the Weizmann Institute of Science, Israel, in 1993. Since 1994, he has been a Ph.D. student with the Department of Computer Science at Technion.

The subject of his thesis is incorporating local shortest path considerations in *Bug*-type motion planning. His research interests include sensor-based motion planning, robot navigation, and task-oriented vision.

Ehud Rivlin (S'90–M'95) received the B.Sc. and M.Sc. degrees in computer science and the M.B.A. degree from the Hebrew University, Jerusalem, and the Ph.D. degree from the University of Maryland, College Park.

Currently, he is an Assistant Professor in the Computer Science Department at Technion, Haifa, Israel. His current research interests include machine vision and robot navigation.