Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots

Ishay Kamon and Ehud Rivlin

Dept. of Computer Science
Technion
Haifa 32000, Israel

Elon Rimon

Dept. of Mechanical Engineering
Technion
Haifa 32000,Israel

Abstract

We present TangentBug, *a new range-sensor based navigation algorithm for two degrees-of-freedom mobile robots. The algorithm combines local reactive planning with globally convergent behavior. For the local planning,* TangentBug *uses the range data to compute a locally shortest path based on a novel structure termed the* local tangent graph, *or* LTG. *The robot uses the LTG for choosing the locally optimal direction while moving towards the target. The robot also uses the* LTG *in its other motion mode, where it follows an obstacle boundary. In this mode the robot uses the* LTG *for making local shortcuts and testing a leaving condition which allows the robot to resume its motion towards the target. We analyze the convergence and performance properties of* TangentBug. *We also present simulation results, showing that* TangentBug *consistently performs better than the classical* VisBug *algorithm. Moreover,* TangentBug *produces paths that in simple environments approach the globally optimal path as the sensor's maximal detection range increases.*

1 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, an autonomous robot cannot base its motion planning on complete apriori 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 (see for example [3, 15]). This approach guarantees that either the target will be reached or the robot will conclude that the goal is unreachable. However, the construction and maintenance of a global model based on sensory information imposes a heavy computational burden on the robot. Recent works also use the global approach to achieve sensor-based navigation of general robots [2, 12], but several implementation issues of these algorithms remain unsolved.

In contrast, local path-planners use the local sensory information in a purely reactive fashion. Local planners typically use navigation vector-fields which directly map the sensor readings to actions (see for example [1, 6]). However, the local approaches do not guarantee global convergence to the target.

Thus, the global approaches suffer from practical implementation problems, while the local ones lack a global convergence guarantee. This paper focuses on a midway approach, originated by Lumelsky and Stepanov [9], which combines local planning with global information. The 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. The algorithms of [9], termed *Bug*1 and *Bug*2, use only position and contact sensors. These algorithms consist of two reactive modes of motion, termed *behaviors*, and transition conditions for switching between them. The two behaviors are moving directly towards the target and following an obstacle boundary. When the robot hits an obstacle it switches from moving towards the target to boundary following. It leaves the obstacle boundary when a *leaving condition*, which ensures that the distance to the target decreases, holds. Different algorithms from this class use different leaving conditions and different data structures [5, 10, 11, 13, 14].

The *Bug* approach minimizes the computational burden on the robot while still guaranteeing global convergence to the target. However, the *Bug* algorithms do not make the best use of the available sensory data to produce short paths. These algorithms use mainly con-

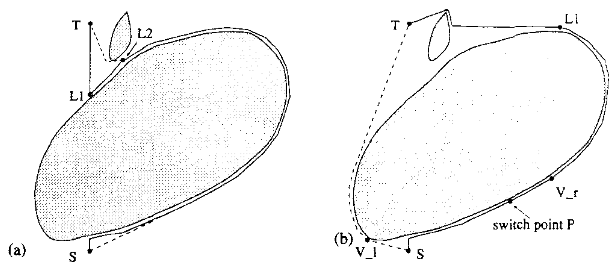

Figure 1. An example comparing *TangentBug* with *VisBug*. (a) The path generated by *VisBug* using contact sensors (solid line), and unlimited sensor range (dashed line). (b) The path generated by *TangentBug* using contact sensors (solid line), and unlimited sensor range (dashed line).

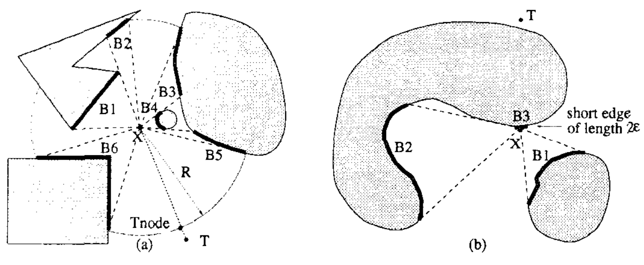

Figure 2. (a) The LTG using detection range R. The sensed obstacles $\mathcal{B}_1, ..., \mathcal{B}_6$ are modeled as thin walls. (b) The LTG when the robot touches a convex obstacle boundary, using unlimited detection range. Note that $\mathcal{B}_2$ and $\mathcal{B}_3$ are distinct sensed obstacles in this case.

tact sensors. Range data is used in *VisBug* [8], but only to find shortcuts to the path generated by *Bug2* (Fig. 1(a)). This paper presents a new algorithm, termed *TangentBug*, which specifically exploits range-data. It constructs a *local tangent graph*, or LTG, and uses it to produce paths which often resemble the shortest path to the goal (Fig. 1(b)).

The paper is organized as follows. Section 2 introduces the LTG. Section 3 presents the *TangentBug* algorithm. Section 4 discusses the convergence and general bounds on the performance of *TangentBug*. Section 5 describes simulation results. These results show that *TangentBug* consistently performs better than the classical *VisBug* algorithm. Moreover, *TangentBug* produces paths that in simple environments approach the globally optimal path as the sensor range increases.

2 The Local Tangent Graph

In this section we review the notion of tangent graph and introduce the range-data based local tangent graph. The tangent graph, denoted *TG*, was introduced in [7]. In a polygonal environment, *TG* is a subgraph of the visibility graph whose vertices are the convex obstacle vertices, together with the start and target points, denoted S and T. The edges of the tangent graph are straight lines that connect the vertices of *TG*, such that the edges lie in the free space and are tangent to the obstacles at their endpoints. Like the visibility graph, the tangent graph contains the shortest path in the environment, but it is significantly smaller. The tangent graph was also generalized in [7] to curved obstacles.

We assume a range sensor which provides perfect readings of the distance to the obstacles within the *visible set.* This is the star-shaped set centered at the current robot location x, whose maximal radius is R. The *local tangent graph*, or LTG, is a tangent graph that includes only the portion of the obstacles which lie in the visible set. The local range data is first divided into distinct *sensed obstacles* at discontinuity points of the range readings (Fig. 2). Each sensed obstacle is then modeled as a thin wall in the real world. This assumption always underestimates the obstacles' size. But a more accurate model would require sophisticated and computationally expensive modeling techniques that we wish to avoid.

The nodes of the LTG are the current robot location x, the endpoints of the sensed obstacles, and optionally an additional node in the direction of the target called T_{node}. If the line segment from x to T is not blocked by any obstacle, T_{node} is at the furthest point on this line within the visible set. The edges of the LTG connect the robot location x with all the other nodes of the LTG. Two examples of the LTG are shown in Fig. 2. As Fig. 2(b) shows, when x lies on a convex obstacle boundary, the boundary is modeled as a short edge of length 2ϵ.

When the sensor range R becomes zero we have the case of a contact sensor, which requires special treatment to fit the above definitions. This case is modeled as distance readings in a small range $R = \epsilon$, where $\epsilon > 0$ is a small scalar (see [4] for more details).

3 The TangentBug Algorithm

The algorithm navigates a point robot in a planar unknown environment populated by stationary obstacles. The sensory input consists of the robot current position x, and the distance from x to the obstacles within a detection range R. First we describe the global structure of the algorithm and then discuss its detailed operation.

TangentBug uses the two basic behaviors of motion towards the target and following an obstacle boundary. In every step the robot constructs the LTG based on its current range readings. It uses the LTG to plan its next action as follows. During motion towards the target, the robot moves in the *locally optimal direction*, which is the direction of the shortest path to the target according to a subgraph of the LTG (the subgraph is described

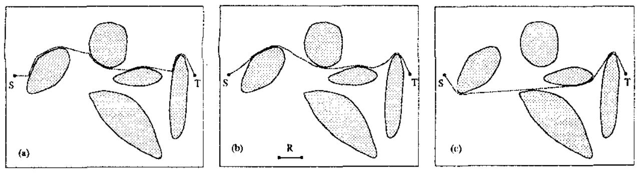

Figure 3. The robot's path in a simple environment using the motion towards the target behavior. (a) Using contact sensors, (b) using limited sensor range R, and (c) using unlimited sensor range. Note the global change in the path caused by the increase in R.

below). By limiting the candidates for the locally optimal direction to a subgraph of the LTG, convergence considerations can be incorporated with shortest-path considerations. In simple scenarios these local decisions will drive the robot along a path that approaches the globally optimal path as R increases (Fig. 3).

Let $d(x,T)$ be the distance from x to T. The robot keeps moving towards the target until it is trapped in the basin of attraction of a local minimum of $d(x,T)$. This situation occurs when an obstacle, located between the robot and the target, generates a local minimum on its boundary. The robot subsequently switches to the boundary following behavior. It chooses a boundary following direction and moves along the boundary while using the LTG to make local shortcuts. But the robot may not leave the boundary before the following *leaving condition* is met. While the robot is following the boundary, it records the minimal distance to the target, $d_{min}(T)$, observed so far along the boundary of the followed obstacle. The robot leaves the obstacle boundary and resumes its motion towards the target when there is a node V in the current LTG which satisfies $d(V,T) < d_{min}(T)$. Here is a summary of the algorithm.

1. Move towards T along the *locally optimal direction* on the current LTG subgraph, until one of the following events occurs:
   - The target is **reached**. Stop.
   - A local minimum is detected. Go to step 2.
2. Choose a boundary following direction. Move along the boundary using the LTG while recording $d_{min}(T)$, until one of the following events occurs:
   - The target is **reached**. Stop.
   - The leaving condition holds: $\exists V \in LTG \ s.t. \ d(V,T) < d_{min}(T)$. Go to step 1.
   - The robot completed a loop around the obstacle. The target is **unreachable**. Stop.

Note that the motion towards the target in step 1 includes both motion between obstacles and sliding along obstacle boundaries (Fig. 3). Next we describe the detailed operation of the algorithm.

During motion towards the target, the robot moves in the direction of the shortest path to the target according to a subgraph $G1$ of the current LTG, $G1 = \{V \in LTG : d(V,T) \leq min(d(x,T), d_{Leave}\}$, where x is the current robot location and d_{Leave} is the guaranteed distance to the target, defined in the last leave point (see below). Initially, when $x = S$, $d_{Leave} = d(S,T)$. The subgraph $G1$ is designed to guarantee that (1) during motion towards the target, $d(x,T)$ is *decreasing* and (2) by the end of the motion-towards-the-target segment, the robot will reach the guaranteed distance $d(x,T) \leq d_{Leave}$. The motion towards the target persists as long as the current $G1$ is non-empty.

Let us describe the construction of the shortest path to the target. Recall that the sensed obstacles are modeled as thin walls and are assumed to be the only obstacles in the environment. We connect to T each node V of the subgraph $G1$, and assign to the edge (V,T) the length of the shortest path from V to T, considering all the visible obstacles. The shortest path from the robot location x to T is then computed on the resulting augmented graph.

The motion towards the target terminates when the robot detects that it is trapped in the basin of attraction of a local minimum of $d(x,T)$. This event occurs when the subgraph $G1$ becomes empty. The robot subsequently terminates its motion towards the target and switches to the boundary following mode.

During boundary-following motion the robot moves away from a local minimum of $d(x,T)$. First note that the local minimum which terminated the motion towards the target is necessarily visible from the robot location, and lies on the boundary of the current *blocking obstacle*, which is the obstacle that exists between the robot and the target. The robot follows the boundary of this obstacle until a leaving condition is satisfied. Let us first describe the initial actions performed by the robot when the boundary-following behavior is activated. The robot chooses a boundary following direction, based on the shortest path to the target according to the LTG [1]. Another initial action is the recording of $d_{min}(T)$ according to the visible portion of the obstacle boundary.

Next the robot starts to follow the obstacle boundary, a process which continues until the leaving condition holds or a loop around the obstacle is completed. During the boundary following motion the robot continuously updates the variable $d_{min}(T)$. It also constructs the LTG and uses it to plan local shortcuts along the followed obstacle boundary. The robot also computes the subgraph $G2 = \{V \in LTG : d(V,T) < d_{min}(T)\}$. This subgraph is usually empty. When it becomes non-

[1]The method for choosing the boundary following direction can be altered without harming the convergence properties.

empty, there exists a node $V \in LTG$ *s.t.* $d(V,T) < d_{min}(T)$. In this case the leaving condition holds, d_{Leave} is set to $d(V,T)$, and the robot switches to motion towards the target. Anticipating the discussion on the convergence of the algorithm, note that by construction $d_{Leave} < d_{min}(T)$. It guarantees that when the new motion-towards-the-target segment ends, it will be due to an obstacle point whose distance to T is shorter than $d_{min}(T)$.

Special treatment is necessary when the robot reaches a sharp corner during a boundary-following mode. The robot's field of view changes discontinuously at this point, and the LTG nodes may change their location discontinuously. This discontinuous change may prevent satisfaction of the leaving condition, and to avoid this problem the robot simulates the following "smoothing" of the corner. Whenever the robot reaches a sharp corner, it varies its motion direction continuously from the initial direction to the final direction, while computing the LTG in all the intermediate directions. As a result, the LTG nodes scan the boundary of the blocking obstacles continuously.

The example in Fig. 1 compares the *TangentBug* algorithm with the *VisBug* algorithm from [8]. The path planned by *VisBug* using contact sensors is shown with solid line in Fig. 1(a) (it is identical to *Bug*2's path). Using unlimited sensor range, *VisBug* plans local shortcuts relative to *Bug*2 path, as shown with dashed line in Fig. 1(a). Using *VisBug*, the robot leaves the obstacle boundary at the point L_2, as soon as the straight line $[L_1, T]$ is visible to the robot. The path planned by *TangentBug* using contact sensors is shown with solid line in Fig. 1(b). The robot switches from motion towards the target to boundary following at the point P, where it detects a local minimum of $d(x,T)$. The robot leaves the first obstacle at the point L_1 where $T_{node} \in$ LTG and $d(T_{node},T) < d_{min}(T)$, and moves towards the target going around the second obstacle until the target is reached. When unlimited sensor range is used, as depicted with dashed line in Fig. 1(b), the resulting path is completely different. The LTG at the starting point S consist of the two endpoints of the blocking obstacle, V_r and V_l. The locally optimal direction is towards the endpoint V_l, and the robot uses the motion towards the target behavior until it reaches the target.

4 Convergence proof

The convergence of *TangentBug* is based on the following ideas. During motion towards the target, the function $d(x,T)$ decreases monotonically and the path length during this mode is guaranteed to be finite. The robot switches to boundary-following mode only at a finite number of points, which are local minima of $d(x,T)$. Moreover, every such local-minimum point is used at most once as a switching point. Every boundary-following segment has finite length since the robot either completes a loop around the obstacle or leaves its boundary before completing a loop. Hence the algorithm terminates after a finite path. If the target is reachable, convergence to the target is guaranteed by the leaving condition. This condition ensures that the robot always terminates its boundary-following mode at some point and resumes its motion towards the target. The last such motion takes the robot to T.

In the following, we consider a point robot in a planar configuration space (*c-space*) populated by a finite number of piecewise-smooth obstacles. The free c-space, denoted $\mathcal{F}$, is the complement of the obstacles' interiors. We also assume that the perimeter of each obstacle is finite. The following lemma is well known.

Lemma 4.1 *The distance function $d(x,T): \mathcal{F} \rightarrow I\!R$ has finitely many local minima in any generic free c-space.*

We now sketch the convergence proof for the case of a contact sensor. The detailed proof, as well as preliminary results for a general range sensor, appear in [4]. We first define several distinguished points along the robot's path. A hit point H_i is a point where the robot first touches an obstacle, a departure point D_i is a point where the moving-towards-the-target behavior drives the robot away from an obstacle boundary (Note, immediately after D_i the robot is still in motion-towards-the-target mode). A switch point P_i is a point where the robot switches from moving-towards-the-target to boundary-following. Using contact sensor, every switch point is also a local minimum point M_i of $d(x,T)$. Last, a leave point L_i is a point where the leaving condition holds and the robot switches from boundary-following to motion towards the target.

The robot's path consists of several (1 to k) motion-towards-the-target segments, each starting from a leave point L_i and ending at a local minimum point M_j. Each segment is divided into subsegments of the following two types. The *direct* subsegments are $[L_i, H_{i+1}]$ and $[D_i, H_{i+1}]$. In these segments the robot moves directly towards the target. The *sliding* subsegments are $[H_i, D_i]$ and $[H_i, M_i]$. In these segments the robot slides along the boundary of a blocking obstacle. The motion-towards-the-target segments are interleaved with boundary-following segments, in which the robot moves from a local minimum point M_i to the next leave point L_i (Fig. 4). Note that $d(x,T)$ decreases along motion-towards-the-target segments, in contrast to boundary-following segments in which $d(x,T)$ may increase.

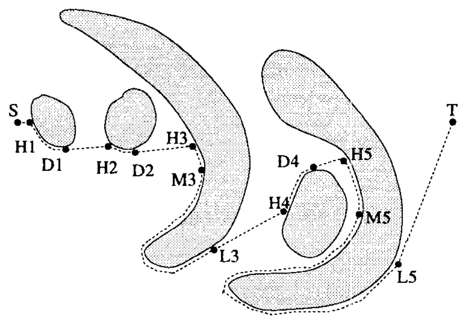

Figure 4. The segments of a *TangentBug* path. The first motion-towards-the-target segment starts at S and ends at M_3. The direct subsegments are $[S, H_1], [D_1, H_2], [D_2, H_3]$, and the sliding subsegments are $[H_1, D_1], [H_2, D_2], [H_3, M_3]$. The robot executes boundary-following motion from M_3 to L_3. The next motion-towards-the-target segment is from L_3 to M_5. The robot follows the obstacle boundary from M_5 to L_5, then moves towards T.

Corollary 4.2 *A motion-towards-the-target segment terminates either at the target T, or at a local-minimum point M_i of $d(x, T)$.*

Lemma 4.3 *Using contact sensor, the path length of every motion-towards-the-target segment is finite.*

Lemma 4.4 *If the target T is reachable from a local-minimum point M_i, the leaving condition will cause the robot to leave the obstacle after a finite-length path.*

Proof: Starting from M_i, the robot uses the boundary-following behavior to move along the obstacle boundary. It will eventually reach a point C which is the closest to T. At this point $d_{min}(T)$ is updated to $d(C, T)$. Since T is reachable, it must be possible to move from C directly towards T. Hence $T_{node} \in$ LTG and $d(T_{node}, T) < d_{min}(T)$, therefore $T_{node} \in G2$ and the robot leaves the boundary. □

Lemma 4.5 *If the robot completes a loop around an obstacle, the target is unreachable.*

Corollary 4.6 *The distance to the target decreases between successive local-minima i.e., $d(M_i, T) > d(M_j, T)$ for $i < j$.*

Proof: While moving from M_i to M_j, the robot follows an obstacle boundary until it reaches a leave point L_i which satisfies $d(M_i, T) \geq d(L_i, T)$. The robot then switches to motion-towards-the-target mode. The leaving condition guarantees that $d(L_i, T) > d_{Leave} \geq d(H_{i+1}, T)$, where H_{i+1} is the next hit point. But the distance $d(x, T)$ decreases during the motion towards the target. Hence $d(H_{i+1}, T) \geq d(M_j, T)$ and it follows that $d(M_i, T) > d(M_j, T)$. □

The following theorem asserts that *TangenBug* always terminates.

Theorem 1 *Using contact sensor, TangenBug always terminates after following a path of finite length.*

Proof: Using contact sensor, switching from motion-towards-the-target to boundary-following takes place at the local minima of $d(x, T)$. According to Corollary 4.6, $d(x, T)$ decreases between successive local-minima along the path. Thus switching to boundary-following occurs at most once at each local-minimum point of $d(x, T)$. According to Lemma 4.1 the number of local minima of $d(x, T)$ is finite. Hence the path consists of finitely many boundary-following segments. Lemma 4.3 and Lemma 4.4 guarantee that the path length for each motion segment is finite. Hence the total path length is finite. □

The following theorem asserts that *TangentBug* is complete.

Theorem 2 *Using contact sensor, TangentBug finds the target if it is reachable from the start point.*

Proof: According to Corollary 4.2, every motion-towards-the-target segment terminates either at the target or at a local-minimum point M_i. When a local minimum M_i is reached, the robot switches to boundary-following mode. If T is reachable from S, Lemma 4.4 guarantees that every boundary-following segment terminates at a leave point L_i. Since the number of boundary following segments is finite and every such segment is followed by a motion-towards-the-target segment, there is a *last* motion-towards-the-target segment. This last segment terminates at the target. □

The following proposition gives an upper bound on the performance of *TangentBug*.

Proposition 4.7 *Using contact sensor, an upper bound L_{max} on the path length that TangenBug generates is:*

$$L_{max} = \|S - T\| + \sum_i \Pi_i + \sum_i \Pi_i \times \sharp Minima_i,$$

where Π_i is the perimeter of the i^{th} obstacle, the summation is over the obstacles that intersect the disc of radius $\|S - T\|$ centered at T, and $\sharp Minima_i$ is the number of local minima of $d(x, T)$ on the i^{th} obstacle boundary.

Proof: The distance $\|S - T\|$ bounds the total path length of the *direct* subsegments of the motion-towards-the-target mode. The term $\sum_i \Pi_i$ bounds the path length of the *sliding* subsegments of the motion-towards-the-target mode. The robot cannot traverse

	world1		world2	
R	Vis_{Bug}	$Tangent_{Bug}$	Vis_{Bug}	$Tangent_{Bug}$
0	1.00	0.77	1.00	0.79
50	0.88	0.75	0.88	0.67
100	0.84	0.74	0.73	0.54
200	0.82	0.73	0.57	0.29
∞	0.81	0.73	0.54	0.24

Table 1. The performance of *TangentBug* algorithm compared to *VisBug* algorithm. Five maximal sensor range values, ranging from 0 to ∞, were tested in two environments. The average path length is presented relative to *VisBug* performance with contact sensors.

the same part of the boundary twice, since the distance to the target $d(x,T)$ decreases during motion towards the target. Hence the robot may hit only those obstacles which intersect the disc of radius $\|S-T\|$ centered at T. Thus $\sum_i \Pi_i$ is an upper bound on the length of the sliding subsegments. As for the length of the boundary-following segments, the robot switches to boundary-following at most once at every local minimum of $d(x,T)$. Moreover, every such local-minimum is used at most once. Hence $\sum_i \Pi_i \times \sharp Minima_i$ is an upper bound on the path length of the boundary-following segments. □

Note that a better bound of $\|S-T\|+2\times\sum_i \Pi_i$ is reported in [13]. The method for achieving this bound, by creating an incremental graph description of the environment, can be incorporated into *TangentBug*.

5 Simulation results

The simulations compare *TangentBug* with the classical *VisBug* in two simulated environments. The simple environment, world1, consists of convex non-intersecting obstacles, while the complex one, world2, consists of concave obstacles with "office-like" shape. Five hundreds randomly chosen start/target points were used in each environment. The results are summed in table 1. The paths produced by *TangentBug* were shorter than the ones produced by *VisBug* in all the scenarios. Increasing the maximal range of the sensors improved the performance of both algorithms. However, *TangentBug* makes better use of the range data.

The simulations also provide the following insight into the dependence of the paths on the sensor range R. In world1, the paths produced by *TangentBug* using unlimited range sensors were optimal in most cases. Only the moving-towards-the-target behavior was used in most of these cases, implying that range data was continuously used for choosing the locally optimal direction. The minor improvement in the path length as the sensor range increases, from 0.77 to 0.73, suggests that *in simple environments small sensor range is sufficient*, because the local data usually leads to the globally correct decisions.

In world2, the effect of the range sensors is more apparent. Significant local shortcuts were performed by scanning the boundaries of concave obstacles, instead of actually following them. This advantage is used by both algorithms. However, *TangentBug* produces shorter paths, since it chooses the boundary following direction based on local information, and it leaves an obstacle boundary much before the line $[S,T]$ is visible.

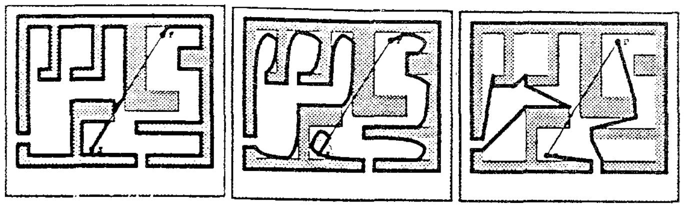
Figure 5. Simulation results of *VisBug* in world2. Left—using contact sensors (path length 1.00). Middle—using limited range sensor 50 (path length 0.89). Right—using unlimited range sensor (path length 0.60).

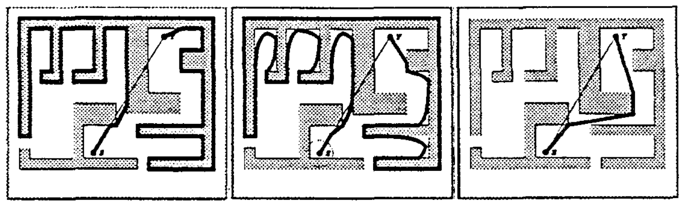
Figure 6. Simulation results of *TangentBug* in world2. Left—using contact sensors (path length 0.79 relative to *Visbug*); Note that the boundary following direction is chosen based on local information. Middle—using limited range sensor 50 (path length 0.70). Right—using unlimited range sensor (path length 0.12).

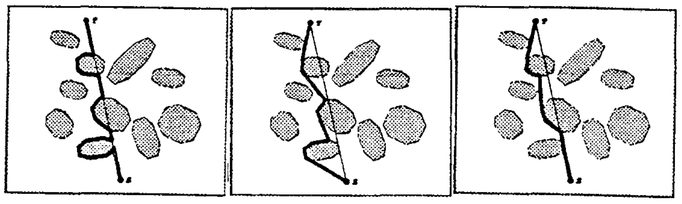
Figure 7. Simulation results in world1. Left—*VisBug* using contact sensors (path length 1.00). Middle - *VisBug* using unlimited range sensor (path length 0.76). Right - *TangentBug* using contact sensors (path length 0.63). Using unlimited range sensors in this case result in minor local shortcuts, and the path length is reduced to 0.6.

6 Concluding Discussion

We presented *TangentBug*, a new range-sensor based globally convergent navigation algorithm for two degrees of freedom mobile robots. We incorporated into the general *Bug* paradigm a notion of the locally shortest path which was based on the tangent graph. The classical tangent graph is defined for a completely known environment. We introduced a local range-data based version of it, termed the *local tangent graph*, or LTG. We reformulated the two basic behaviors of the *Bug* family in a way which continuously uses the LTG. We also defined new transition conditions for switching between the two behaviors.

TangentBug uses the range data for choosing the locally optimal direction while moving towards the target, for making local shortcuts while following an obstacle boundary, and for testing the leaving condition which allows the robot to leave an obstacle boundary. The simulation results showed improvement (from 0.79 to 0.24 in the complex environment) in the average path length as the sensor detection range increased. The simulation also indicated a significant advantage of *TangentBug* relative to the classical *VisBug* algorithm in all the tested scenarios.

TangentBug combines purely reactive decisions with globally convergent behavior in a way which minimizes the need for global positioning. It requires global positioning only for the detection of the target and completion of a loop around an obstacle boundary. This combination of purely reactive behavior with minimal global positioning makes *TangentBug* especially simple to implement.

Using *TangentBug*, additional information can be easily incorporated to improve performance in practical scenarios. For example, the LTG can be improved when accumulated sensory data, e.g. using occupancy grids, is used to increase both sensor reliability and the effective sensor range. Another way to improve performance is by modifying the expected path length from the LTG nodes to the target, based on a nominal global model of the environment, or more informative sensing modalities. However, more informative sensing modalities are currently not readily available, and more sophisticated modeling techniques would increase the computational burden on the robot.

References

[1] J. Borenstein and Y. Koren. Real time obstacle avoidance for fast mobile robots in cluttered environments. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 572–577, 1990.

[2] H. Choset and J. W. Burdick. Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. In *IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 1995.

[3] G. Foux, M. Heymann, and A. Bruckstein. Two dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Transactions on Robotics and Automation*, 9(1):96–102, 1993.

[4] I. Kamon, E. Rimon, and E. Rivlin. A new range-sensor based globally convergent navigation algorithm for mobile robots. CIS - Center of Intelligent Systems 9517, Computer Science Dept., Technion, Israel, 1995.

[5] I. Kamon and E. Rivlin. Sensory based motion planning with global proofs. In *Proceedings of the Conference on Intelligent Robots and Systems, IROS*, pages 435–440, 1995.

[6] 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.

[7] Y. H. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *International Journal of Robotic Research*, 11(4):376–382, 1992.

[8] 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.

[9] 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.

[10] 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.

[11] H. Noborio. A sufficient condition for designing a family of sensor based deadlock free path planning algorithms. *Advanced Robotics*, 7(5):413–433, 1993.

[12] E. Rimon and J. F. Canny. Construction of c-space raodmaps from local sensory data. what should the sensors look for ? In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 117–123, 1994.

[13] A. Sankaranarayanan and M. Vidyasagar. A new path planning algorithm for a point object moving amidst unknown obstacles in a plane. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1930–1936, 1990.

[14] 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 *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1734–1941, 1991.

[15] A. Stentz. Optimal and efficient path planning for partially known environments. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 3310–3317, 1994.