Ishay Kamon*
Elon Rimon†
Ehud Rivlin*

*Department of Computer Science
†Department of Mechanical Engineering
Technion
Haifa 32000, Israel

TangentBug: A Range-Sensor-Based Navigation Algorithm

Abstract

The Bug family algorithms navigate a 2-DOF mobile robot in a completely unknown environment using sensors. TangentBug is a new algorithm in this family, specifically designed for using a range sensor. TangentBug uses the range data to compute a locally shortest path, based on a novel structure termed the local tangent graph (LTG). The robot uses the LTG for choosing the locally optimal direction while moving toward the target, and for making local shortcuts and testing a leaving condition while moving along an obstacle boundary. The transition between these two modes of motion is governed by a globally convergent criterion, which is based on the distance of the robot from the target. We analyze the properties of TangentBug, and present simulation results that show that TangentBug consistently performs better than the classical Bug algorithms. The simulation results also show that TangentBug produces paths that in simple environments approach the globally optimal path, as the sensor's maximal detection-range increases. The algorithm can be readily implemented on a mobile robot, and we discuss one such implementation.

1. Introduction

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

In the global sensor-based planning approach, the mobile robot builds a global model of the environment based on sensory information, and uses it for path planning (Foux, Heymann, and Bruckstein 1993; Stentz 1994). Some recent work also uses a global model for sensor-based navigation of general robots (Choset and Burdick 1995; Rimon 1997), but it too has so far been implemented only on mobile robots. The global planning 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. Moreover, the reliance on a global model for the navigation requires frequent localization of the robot relative to the model, a process that is difficult, due to the inherent uncertainties of practical sensors (Crowley and Demazeau 1993; Leonard and Durrant-Whyte 1992; Rencken 1993).

In contrast, local path planners use local sensory information in a purely reactive fashion. In every control cycle, the robot uses its sensors to locate nearby obstacles and to plan its next action based on this local information. The local planners are usually much simpler to implement than the global ones, since they typically use navigation vector fields that directly map the sensor readings to actions. Different methods are used to choose or learn these vector fields, including the potential-field method and its variations (Arkin 1987; Khatib 1985), fuzzy logic approaches (Goodridge and Luo 1994; Reignier 1994), and methods that construct specialized data structures to make more reliable decisions (Bauer, Feiten, and Lawitzky 1993; Borenstein and Koren 1990). However, while the local approaches are simple to implement, they dc not guarantee global convergence to the target. A local planner may get trapped in a local minimum or in a loop. Thus the global approaches are difficult to implement, while the local ones lack a global convergence guarantee.

We focus on a midway approach, called the *Bug* approach which was originated by Lumelsky and Stepanov (1987), an has been subsequently studied by several others (Lumelsk 1991; Noborio and Yoshioka 1993; Sankaranarayanan an Vidyasagar 1991). This approach combines local plannin with a globally convergent criterion, as follows. It uses tw reactive modes of motion, and two transition conditions fc

The International Journal of Robotics Research
Vol. 17, No. 9, September 1998, pp. 934–953,
©1998 Sage Publications, Inc.

switching between these two modes. The modes are (1) motion toward the target, and (2) following an obstacle boundary. The robot initially moves toward the target. When it hits an obstacle, it switches to boundary-following mode. It leaves the obstacle boundary when a *leaving condition*, which monitors a globally convergent criterion, holds. The leaving condition ensures that the distance to the target decreases at successive hit points, and thus guarantees global convergence to the target.

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. Thus, the approach minimizes the computational burden on the planner while still guaranteeing global convergence to the target. However, with the exception of VisBug (Lumelsky and Skewis 1990), the Bug algorithms were designed for using contact sensors, and not range (or distance) sensors. Even VisBug uses the range data only to find shortcuts relative to the path generated by the classical Bug2 algorithm (Lumelsky and Stepanov 1987), which relies on contact sensors (Fig. 1a). We present a new Bug algorithm, termed TangentBug, that specifically exploits range data. The new algorithm uses the notion of a *tangent graph* to construct a *local tangent graph*, which is used to produce paths that often resemble the shortest path to the goal (Fig. 1b).

The next section reviews the tangent graph and introduces the local tangent graph. We then present the TangentBug algorithm as a one-parameter family of algorithms, parameterized by the maximal detection range R, where $0 \leq R \leq \infty$. In Section 4 we show that TangentBug is globally convergent, and discuss general bounds on the performance of the algorithm. In Section 5 we present simulation results that show that TangentBug often generates paths that approach the shortest path as detection range R increases. The simulations also compare TangentBug with the classical VisBug algorithm, showing that TangentBug generates significantly shorter paths in congested, officelike scenarios. Finally, we discuss our implementation of TangentBug on a mobile robot.

2. The Tangent Graph

In this section we review the tangent graph, which requires global knowledge of the environment. Then we introduce its sensor-based local version, called the local tangent graph.

2.1. Review of the Global Tangent Graph

First, recall the definition of the *visibility graph*. Consider a polygonal environment in which a start point, S, and a target point, T, are specified. The visibility graph, denoted $VG(V_v, E_v)$, is the graph whose vertices V_v are the obstacle vertices and the points S and T, and whose edges E_v are the collision-free line segments that connect the graph vertices. An example of the visibility graph for an environment with three obstacles is shown Figure 2a. It is known that the shortest collision-free path from S to T always lies on the visibility graph. Thus, an optimal collision-free path from S to T can always be found by limiting the search to the visibility graph. However, the visibility graph contains many edges that never participate in the shortest path. For the purpose of comparison with the tangent graph, if there are N obstacle vertices, the number of edges in the visibility graph is bounded by $\|E_v\| \leq N^2$.

The tangent graph, denoted $TG(V_t, E_t)$, was introduced in Rohnert (1986) and extended by Liu and Arimoto (1992). It is the subgraph of VG obtained by retaining only the convex vertices of the obstacles, and the edges which are bitangent to these vertices (Fig. 2b). Like the visibility graph, the tangent graph contains the shortest path in the environment. But the tangent graph is the *minimal* such graph, as the removal of any of its edges would destroy its optimality. The tangent graph has a significantly smaller number of vertices and edges than the visibility graph, as Figure 2 shows. Thus, searching for the shortest path is more efficient on the tangent graph. To estimate the size of TG, assume that the obstacles are the union of M convex polygons, such that the boundary curves of any two overlapping convex polygons intersect at two points. (This condition can always be met by suitable subdivision of the polygons.) It is known that any two convex polygons can have at most four bitangent edges. Hence the number of edges in TG is bounded by $\|E_t\| \leq 2M^2 + N$. Since M is typically much smaller than N, we see that $\|E_t\|$ is $O(N)$, while $\|E_v\|$ is $O(N^2)$.

REMARK 1. The tangent graph was also generalized in Liu and Arimoto (1992) to piecewise-smooth obstacles. In this case, the boundary of each obstacle is partitioned into convex and concave arcs. The edges of the tangent graph are the convex boundary arcs and the line segments that are bitangent to convex boundary arcs.

2.2. The Local Tangent Graph

We assume a range sensor that provides perfect readings of the distance to the obstacles within the *visible set*, defined as follows. Let the free space be the complement of the obstacles' interior, and let x be the current robot location. Then the visible set is the star-shaped set contained in the free space, with center at x and maximal radius R. The *local tangent graph*, or LTG, is a tangent graph that includes only the portion of the obstacles that lie in the visible set. The LTG is constructed as if the local range information represents all the obstacles in the environment. This assumption allows the robot to base its local decisions only on the currently observable obstacles, thereby greatly simplifying the data processing. The local-range data is first divided into distinct *sensed obstacles* by a process described below. Each sensed obstacle is then modeled as a thin wall in the real world. The thin-wall model always underestimates the obstacles' sizes, but a more accurate model would require computationally expensive modeling techniques that we wish to avoid.

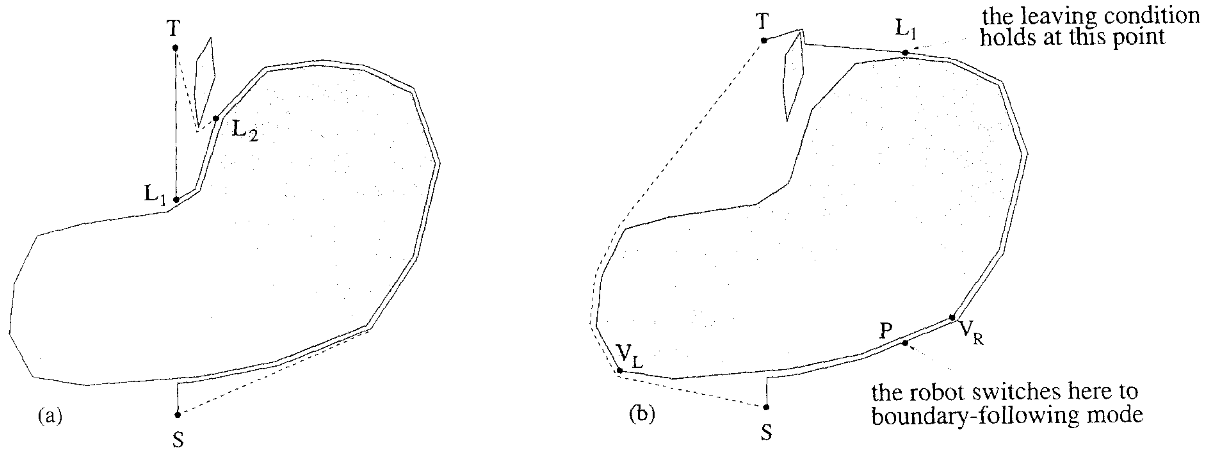
T
L2
L1
(a)
S
T
L1
the leaving condition holds at this point
VL
P
VR
(b)
S
the robot switches here to boundary-following mode

Fig. 1. An example comparing TangentBug with VisBug: (a) the path generated by VisBug using a contact sensor (solid line), and a sensor with unlimited detection range (dashed line); (b) the path generated by TangentBug using the same sensors.

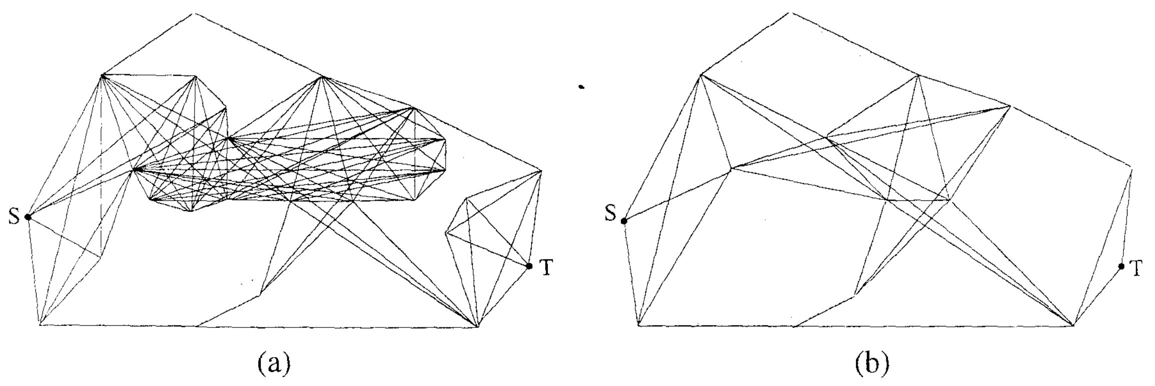

Fig. 2. A comparison of (a) the visibility graph and (b) the tangent graph.

Next, we describe the partitioning of the range data into sensed obstacles. The range information can be represented as a function $r(\theta)$, where θ is the angle relative to a predefined direction and $r(\theta)$ is the distance in the direction θ from the robot location x to the nearest point on an observed boundary. The range data is divided into distinct sensed obstacles based on the fact that the range changes continuously along the boundary of a single visible part of an obstacle. The function $r(\theta)$ has finitely many discontinuities, which occur on tangent lines or at angles where an obstacle is occluded. A continuous range interval $(r(\alpha), r(\beta))$, whose endpoints are either discontinuity points or points where $r(\theta) = R$, is considered a distinct sensed obstacle. Note that a single real obstacle can generate several sensed obstacles, as shown in Figure 3a. Last, when the robot touches an obstacle edge, the portion of the edge within the visible set is considered to be a sensed obstacle, as shown in Figure 3b.

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 an obstacle within the visible set, T_{node} is at the furthest point on this line. In particular, T_{node} is at T if the target lies in the visible set. The edges of the LTG connect the robot location x with all the other nodes of the LTG. The LTG additionally contains edges that are bitangent to the sensed obstacles, but the algorithm never uses these edges and they are not explicitly constructed Two examples of the LTG are shown in Figure 3.

A detection range of $R = 0$ corresponds to contact sensors and this case requires special treatment to fit the above def initions. We assume that the contact sensors can determin if there is free space in the direction of the target, and ca recover the local orientation of a touched boundary. Thes two types of information are modeled as distance readings i a small range of $R = \epsilon$, where $\epsilon > 0$ is a small paramete If there is free space in the direction of T, it is represente by placing T_{node} at a distance ϵ from x in the direction of T The boundary touched by the robot is represented by a sho

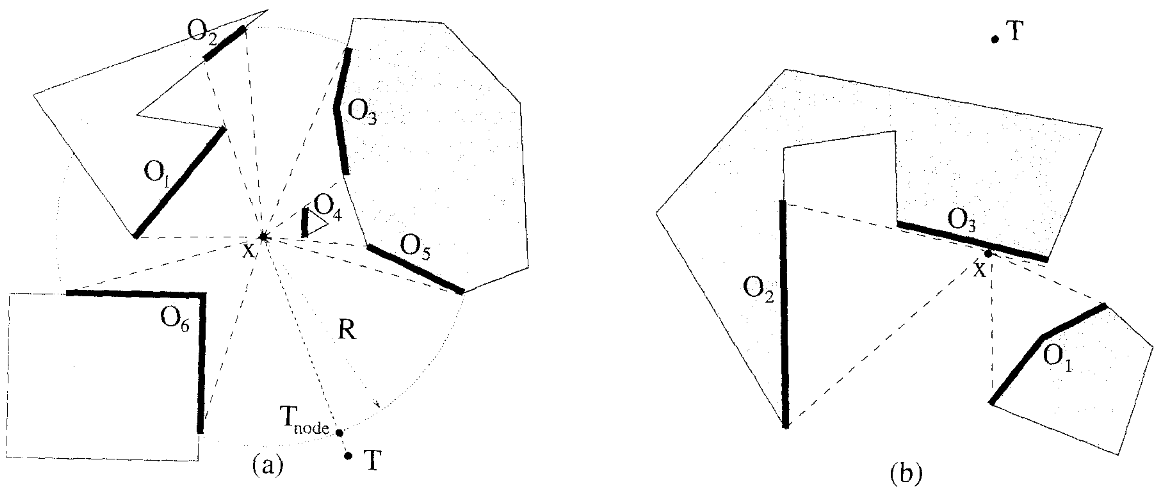

Fig. 3. (a) Using a detection range R, the LTG edges connect the robot location x with the endpoints of the sensed obstacles and the T_{node} (bitangent edges are not shown). (b) The LTG when the robot touches an obstacle edge.

edge of length 2ϵ tangent to the boundary, and the endpoints of this edge become vertices of the LTG.

REMARK 2. The LTG can also be used for general piecewise-smooth obstacles, with the following additional construction. When the robot touches a convex obstacle arc, the visible set locally intersects the boundary only at the contact point. We represent this boundary by a short segment of length 2ϵ tangent to the boundary, and make the endpoints of this segment nodes of the LTG.

3. The TangentBug Algorithm

The TangentBug algorithm navigates a point robot in a planar unknown environment populated by stationary obstacles. The obstacles can have any piecewise-smooth boundary, although in the sequel we focus on polygonal obstacles. The sensory input during the navigation consists of the robot's 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 we discuss its detailed operation.

3.1. Algorithm Description

TangentBug uses two basic behaviors: motion toward the target and obstacle boundary-following. In every step, the robot constructs the LTG based on the current range readings, and uses the LTG as follows. During motion toward 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 described below. Let the function $d(w, T)$ measure the Euclidean distance of a point w from T, such that the points w belong to the free space. The robot keeps moving toward the target until it is trapped in the basin of attraction of a local minimum of $d(w, T)$. Then it switches to the boundary-following behavior.

The robot 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 when there is a node V_{leave} in the current LTG that satisfies $d(V_{leave}, T) < d_{min}(T)$. After leaving the obstacle, the robot performs a transition phase before it resumes its motion toward the target. In the transition phase, the robot moves directly toward V_{leave} until it reaches a point Z that satisfies $d(Z, T) < d_{min}(T)$. At this point the robot resumes its motion toward the target. A summary of the algorithm now follows.

1. Move toward 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: $\exists V_{leave} \in LTG$ $s.t.$ $d(V_{leave}, T) < d_{min}(T)$. Go to step 3.

- the robot completed a loop around the obstacle. The target is **unreachable**. Stop.

3. Perform the transition phase. Move directly toward V_{leave} until reaching a point Z that satisfies $d(Z, T) < d_{min}(T)$. Go to step 1.

3.2. Motion toward the Target

During motion toward the target, the robot moves in the direction of the shortest path to the target according to a subgraph of the current LTG, which is defined as follows. Consider an LTG edge that emanates from the current robot location. We call such an edge *admissible* if the motion of the robot along the edge decreases its distance to the target. The subgraph G_1 consists of the admissible LTG edges and the LTG nodes associated with these edges. Thus the nodes of G_1, denoted V_{G_1}, are given by $V_{G_1} = \{x\} \cup \{V \in \text{LTG} : (V - x) \cdot (T - x) > 0\}$, where x is the current robot location. By limiting the computation to the subgraph G_1, we are able to incorporate local shortest-path considerations with global convergence considerations.

The shortest path to the target is constructed as follows. Recall that the visible obstacles are modeled as thin walls, and are assumed to be the only obstacles in the environment. The algorithm adds a *virtual edge* from each node V of G_1 to T, and assigns to the edge the length of the shortest path from V to T based on the currently visible obstacles. The shortest path from the robot location x to T is then computed on the resulting *augmented graph*. Figure 4 shows two examples of the construction. In Figure 4a, a detection range R is used, and the subgraph G_1 is identical to the LTG. The augmented graph has two additional edges from the nodes V_L and V_R to T. In Figure 4b, an unlimited detection range is used, and G_1 consists of the nodes V_L, V_R, and the node generated from occlusion B_L. The augmented graph consists of two virtual edges connecting the nodes B_L and V_R to T. Note that the shortest path from V_R to T, according to the visible obstacles, is not a straight-line segment.

The motion toward the target terminates when the robot detects that all the nodes V in the subgraph G_1 satisfy $d(V, T) \geq d(x, T)$. As we show in the ensuing analysis, this condition occurs when the robot is trapped in the basin of attraction of a local minimum of the distance function $d(w, T)$. The robot subsequently terminates its motion toward the target and switches to the boundary-following mode. Several examples of this event are illustrated in Figure 5, including an example where the switch occurs at the start point S.

3.3. Following an Obstacle Boundary

During the boundary-following motion, the robot moves away from the local minimum of $d(w, T)$, which terminated the motion toward the target. The local minimum necessarily lies on the boundary of some obstacle, called the *blocking obstacle*. Moreover, the local minimum is necessarily visible from the robot's location. The robot first chooses a direction along which to follow the boundary of the blocking obstacle, based on the shortest path to the target, according to the LTG.[1] Then it records in the variable $d_{min}(T)$ the minimal distance to the target observed along the visible portion of the blocking obstacle.

Next, the robot follows the obstacle boundary. During this motion, the robot continuously updates the variable $d_{min}(T)$, and uses the current LTG to plan local shortcuts along the followed obstacle boundary. The robot also computes the subgraph $G_2 = \{V \in \text{LTG} : d(V, T) < d_{min}(T)\}$. This subgraph is usually empty. When it becomes nonempty, there exists a node $V_{leave} \in \text{LTG}$ such that $d(V_{leave}, T) < d_{min}(T)$. (If G_2 contains several nodes, V_{leave} is chosen on the shortest path to T in the augmented graph, based on G_2.) When V_{leave} is detected, the leaving condition holds and the robot leaves the obstacle boundary.

After leaving the obstacle, the robot performs a transition phase before it resumes its motion toward the target. In this phase the robot moves directly toward V_{leave} until it reaches a point Z that satisfies $d(Z, T) < d_{min}(T)$. The transition phase ensures that when the next motion-toward-the-target segment terminates, it will be at a point whose distance to T is shorter than $d_{min}(T)$. This is one of the key properties that guarantee convergence of the robot to the target.

Finally, special treatment is necessary when the robot reaches a convex obstacle vertex during the boundary-following mode of motion. (This event can occur for any detection range.) The robot's field of view changes discontinuously at this point, and the LTG nodes may change their location discontinuously. Such a discontinuous change may cause the robot to miss a suitable leave point. To prevent this possibility, the robot simulates the following "smoothing" of the vertex. Whenever the robot reaches an obstacle vertex, it assumes that the boundary's tangent vector changes continuously from the direction of the edge entering the vertex to the direction of the edge emerging from the vertex. The robot then computes the LTG for all the intermediate directions, and as a result the LTG nodes continuously scan the boundary of the obstacles visible from the vertex (Fig. 14).[2]

3.4. An Example

Figure 1 illustrates the paths generated by TangentBug and the classical VisBug algorithms (Lumelsky and Skewis 1990). The path planned by VisBug using a contact sensor is shown

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

2. The leaving condition can be tested more efficiently, without computing multiple LTGs, as follows. Upon arriving to a convex vertex, the robot computes the LTG, updates $d_{min}(T)$, and detects the obstacle boundary that blocks the way to the target. The robot then searches this boundary for a point C that satisfies $d(C, T) < d_{min}(T)$. If such a point C exists, the leaving condition holds and V_{leave} is set to C.

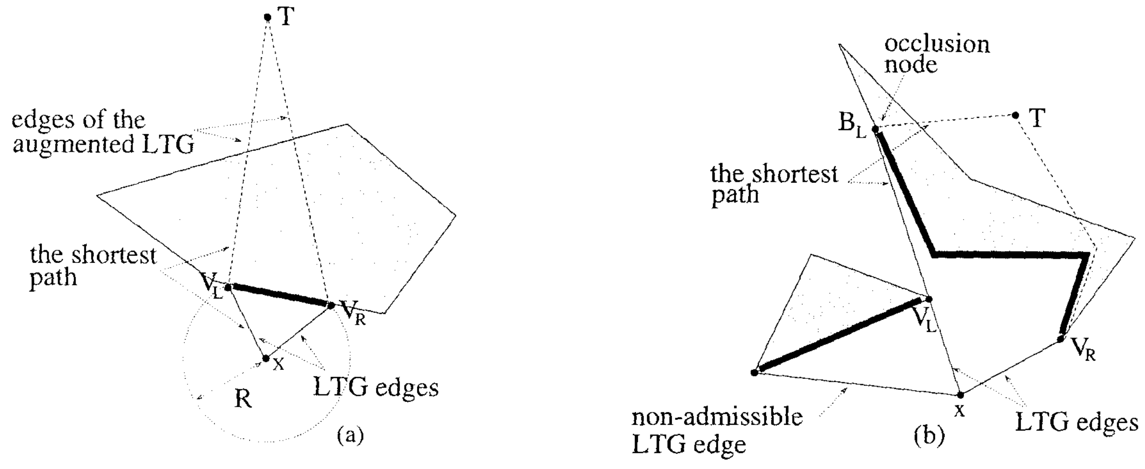

Fig. 4. The subgraph G_1 and the augmented graph for (a) a detection range R, and (b) an unlimited detection range.

with solid line in Figure 1a (it is identical to Bug2's path). Using unlimited sensor range, VisBug plans local shortcuts relative to Bug2's path, as shown with dashed line in Figure 1a. 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 a contact sensor is shown with solid line in Figure 1b. The robot switches from motion toward the target to boundary-following at the point P, where it detects a local minimum of $d(w, T)$. The robot leaves the first obstacle at the point L_1 where $T_{node} \in$ LTG and $d(T_{node}, T) < d_{min}(T)$. When unlimited sensor range is used, as depicted with a dashed line in Figure 1b, the resulting path is completely different. The LTG at S consists of the two endpoints of the blocking obstacle, V_L and V_R, and the locally optimal direction is toward V_L. In this case, the robot continuously uses the motion toward the target behavior until it reaches the target. A more detailed comparison of TangentBug with VisBug is carried out in Section 5.

4. Algorithm Analysis

We investigate several properties of TangentBug, making the following assumptions. The mobile robot is modeled as a point moving in a planar configuration space (C-space). We assume that the point representing the robot moves in a bounded region of the plane, which is populated by stationary polygonal obstacles. The free C-space, denoted $\mathcal{F}$, is the complement of the obstacles' interiors. Note that $\mathcal{F}$, being bounded and polygonal, contains finitely many local minima of the distance function $d(w, T)$. Last, we consider the two extreme types of range sensors: a contact sensor and a sensor with unlimited detection range.

We define several distinguished points along the path. A switch point P_i is a point where the robot switches from motion toward the target to boundary-following. A local-minimum point M_i is the local minimum of $d(w, T)$ associated with the switch event at P_i. A leave point L_i is a point where the leaving condition holds and the transition phase begins. Last, a transition point Z_i is a point where the transition phase terminates and the motion toward the target resumes. We need the following property of the motion-toward-the-target mode.

LEMMA 1. The distance of the robot from the target, $d(x, T)$, decreases monotonically during each motion-toward-the-target segment, and between successive motion-toward-the-target segments.

Proof. By definition of the algorithm, during motion toward the target the robot moves along admissible edges, along which $d(x, T)$ decreases. The distance $d(x, T)$ also decreases between successive motion-toward-the-target segments, since the leaving condition guarantees that $d(P_i, T) \geq d_{min}(T) > d(Z_i, T)$, where P_i is the endpoint of the previous segment and Z_i is the start point of the next segment. □

4.1. Using a Contact Sensor

When a contact sensor is used, the LTG consists of the following two types of nodes. If there is free space in the direction of T, it is represented by T_{node} which is placed at a distance ϵ from x in the direction of T. If the robot touches an obstacle boundary, the boundary is represented by a short edge of length 2ϵ tangent to the boundary, and the endpoints of this edge become nodes of the LTG. For this construction the transition phase is of length ϵ, and we may assume that $L_i = Z_i$. Further, when a contact sensor is used, the robot switches from motion toward the target to boundary-following when it reaches a local-minimum point M_i. Hence all the switch points satisfy $P_i = M_i$.

Each motion-toward-the-target segment consists of the following two types of subsegments. Let H_i be a point where

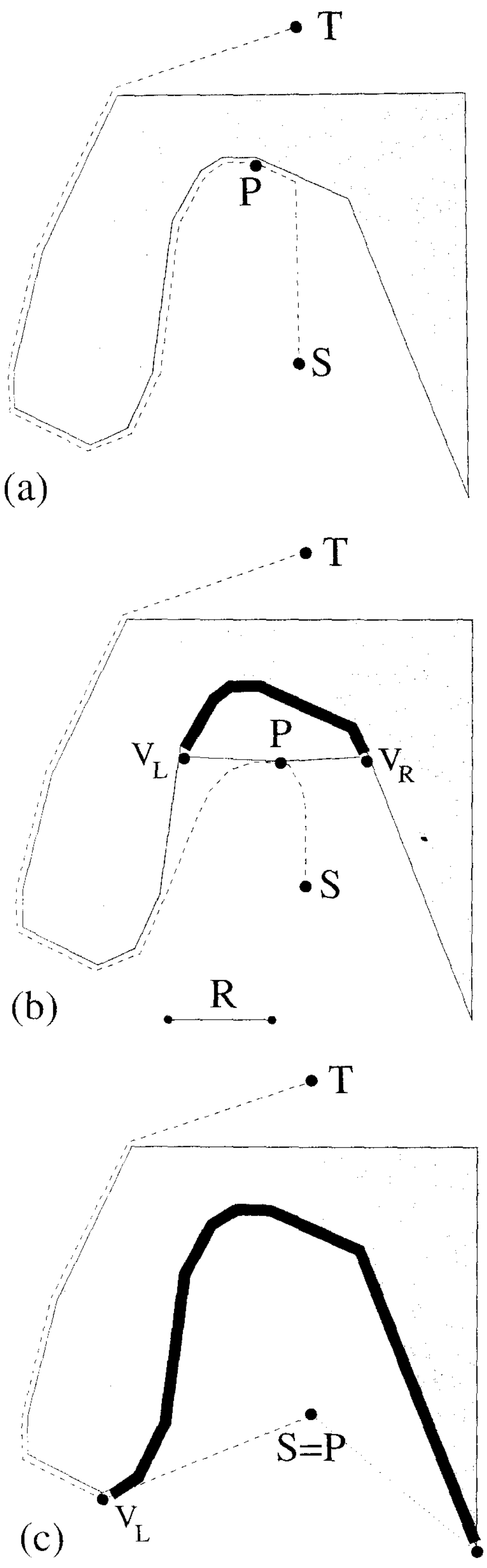

Fig. 5. Starting at S, the robot terminates its motion toward the target at P, where it detects a local minimum. The robot next switches to boundary-following. The point P is shown for (a) a contact sensor, (b) a limited sensor range R, and (c) an unlimited sensor range.

the robot first hits an obstacle, and let D_i be a point where the robot departs from an obstacle, still in the motion-toward-the-target mode. Then the *direct* subsegments are $[L_i, H_{i+1}]$ and $[D_i, H_{i+1}]$. In these subsegments, the robot moves directly toward the target. The *sliding* subsegments are $[H_i, D_i]$ and $[H_i, M_i]$. In these subsegments, the robot slides along the boundary of an obstacle, still in the motion-toward-the-target mode. The motion-toward-the-target segments are interleaved with boundary-following segments, in which the robot moves from a local minimum point M_i to a leave point L_i (Fig. 6). A detailed analysis of TangentBug for the contact-sensor case appears in Kamon, Rimon, and Rivlin (1995). However, of special interest is the following upper bound on the performance of TangentBug. In the proposition, D_T denotes the disc with center at T and radius $\|S-T\|$.

PROPOSITION 1. Using a contact sensor, an upper bound L_{max} on the path length of TangentBug is

$$L_{max} = \|S-T\| + \sum_{i \in \mathcal{I}} \Pi_i \times \sharp\,\text{Minima}_i,$$

where $\mathcal{I}$ is the index set of the obstacles that intersect the disc D_T, Π_i is the perimeter of the ith obstacle, and $\sharp\,\text{Minima}_i$ is the number of local minima of $d(w, T)$ in D_T along the ith obstacle boundary.

Proof. First, consider the motion toward the target segments. Using Lemma 1, the start point of each direct subsegment is closer to T than the endpoint of the previous direct subsegment, and these subsegments all point toward T. Hence the total length of the direct subsegments is bounded by $\|S-T\|$. Lemma 1 also implies that during motion toward the target, the robot hits obstacles only inside the disc D_T, and the robot traverses any boundary segment at most once. Hence the total length of the sliding subsegments is bounded

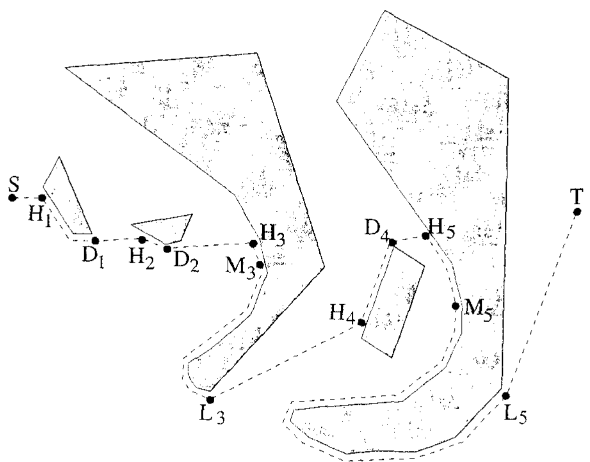

Fig. 6. In the motion-toward-the-target segment from S to M_3, the direct subsegments are $[S, H_1]$, $[D_1, H_2]$, and $[D_2, H_3]$. The sliding subsegments are $[H_1, D_1]$, $[H_2, D_2]$, and $[H_3, M_3]$.

by the sum $\sum_{i \in \mathcal{I}} \Pi_i$. Consider now the boundary-following segments. In Kamon, Rimon, and Rivlin (1995), we show that the robot switches to boundary-following at most once at every local minimum of $d(w, T)$. Moreover, only the local minima in D_T are used. Hence $\sum_{i \in \mathcal{I}} \Pi_i \times \sharp\,\text{Minima}_i$ is an upper bound on the length of the boundary-following segments. Last, if T is reachable from S the global minimum on each obstacle never initiates boundary-following; hence we subtract unity from $\sharp\,\text{Minima}_i$. □

REMARK 3: The proposition specifies an upper bound on the path length of TangentBug. But we may also ask what is the lower bound. Lumelsky and Stepanov (1987) showed that the worst-case[3] lower bound on the path length of any Bug-type algorithm that uses a contact sensor is $\|S - T\| + \sum_{i \in \mathcal{I}} \Pi_i$. (Sankaranarayanan and Vidyasagar [1991] provided a more refined analysis of the lower bound.) However, in practice, the average performance is more important, and we compare the average performance of TangentBug with the classical VisBug algorithm in Section 5.

4.2. Using a Range Sensor

We now investigate the case of a range sensor with unlimited detection range. First, we show that TangentBug terminates after a finite-length path. Then we show that TangentBug is complete, and discuss the upper bound on the path length. We begin with several definitions. For unlimited detection range, the LTG nodes are the current robot location x, possibly the target T, and the endpoints of the visible obstacles. The latter can be of the following two types. The first, called *tangent nodes*, are convex obstacle vertices V, with the property that the line containing the segment $[x, V]$ is tangent to the obstacle at the vertex V. The second, called *occlusion nodes*, are points in the interior of obstacle edges generated from occlusion by convex obstacle vertices (Fig. 4). At every step the robot moves in the direction of a particular tangent node on the LTG. This node is called the current *focus node*, and is denoted F.

Our first goal is to show that TangentBug terminates. The following lemma asserts that TangentBug generates a piecewise-linear path. Each change of direction in this path is associated with a change of the current focus node, which can occur in one of the following three ways. In the event called *type 1*, the robot reaches F and chooses a new focus node. In the *type 2* event, the shortest path to T on the augmented graph ceases to pass through F, and a new focus node is selected. In the event called *type 3*, the edge of the current focus node ceases to be admissible, and the edge with its endpoint F is removed from the subgraph G_1.

3. That is, for any Bug-type algorithm that uses a contact sensor, there exists an obstacle course where the algorithm generates a path at least that long.

LEMMA 2. Using unlimited sensor range, TangentBug generates a piecewise-linear path.

Proof. First, consider the motion-toward-the-target mode. During this mode of motion, the robot moves toward the current focus node F. The focus node remains the same obstacle vertex until an event of the three types mentioned above occurs. We show in the sequel that these events occur only at a discrete set of points along the robot's path. Hence the path is piecewise linear in this mode of motion. Next, consider the boundary-following mode of motion. The boundary-following direction is fixed, and the motion direction changes only at the obstacle vertices. Hence the robot's path during boundary-following is also piecewise linear. Finally, during the transition phase the robot moves in a straight line from a leave point L_i to the transition point Z_i. □

We now consider separately the number of direction changes associated with the three types of events, starting with the type 1 events.

LEMMA 3. During motion toward the target, there are finitely many direction changes due to type 1 events.

Proof. According to Lemma 1, the distance of the robot from the target, $d(x, T)$, decreases monotonically along motion-toward-the-target segments. Thus the robot may visit any obstacle vertex at most once. Since type 1 events occur when the robot reaches a convex obstacle vertex, the number of these events is finite. □

Next, we show that the number of direction changes associated with type 2 events (a change of focus node due to shortest-path consideration) is finite. We first characterize the locally shortest path. Recall that the augmented graph is the subgraph G_1, augmented with virtual edges from the nodes of G_1 to T. Recall, too, that the length of a virtual edge from a node V is the length of the shortest path from V to T, based on the currently visible obstacles. Suppose now that one of the visible obstacles lies between the robot and the target. This obstacle, called the *blocking obstacle*, induces two shortest-path candidates on the augmented graph. The first, called the *left shortest path*, is the shortest path from x to T that circumvents the blocking obstacle from its left side. The second, called the *right shortest path*, is the shortest path that circumvents the blocking obstacle from its right side. We use the following notation. The first node on the left and right paths is denoted V_L and V_R, respectively. The left and right endpoints of the blocking obstacle are always nodes of the LTG, and these nodes are denoted B_L and B_R. We need the following property of the shortest path.

LEMMA 4. When a blocking obstacle exists, the augmented graph contains a left path if and only if the edge that connects x with B_L is admissible. In that case, the left shortest path passes through B_L. A similar result holds for the right shortest path, which passes through B_R.

Proof. Consider the left path. It can be verified that the shortest path to T, considering only the visible obstacles, always passes through an endpoint of the blocking obstacle (Kamon, Rimon, and Rivlin 1995). Hence if the edge $[x, B_L]$ is admissible, the left shortest path must pass through this edge. If the edge $[x, B_L]$ is not admissible, the angle α between the edge $[x, B_L]$ and the line segment $[x, T]$ satisfies $\alpha \geq 90°$. It follows that all the LTG edges to the left of $[x, T]$ are also nonadmissible, since the angles between these edges and the line $[x, T]$ exceed 90°. Hence there is no left path on the augmented graph. A similar argument holds for the right path. □

The next lemma characterizes the ways in which type 2 events can occur.

LEMMA 5. The events of type 2 occur only when an LTG node, which is either a tangent or an occlusion node, is removed or added to the LTG.

Proof Sketch. In the proof (which appears in the appendix), we consider two cases. In the first case, the robot moves from X_0 to X_1 in a way that keeps all the LTG nodes fixed. As illustrated in Figure 7a, suppose that at X_0 the left path is shorter than the right path. The left and right paths from X_1 pass through the same nodes V_L and V_R. This, together with an application of the triangle inequality to the nodes X_0, X_1, V_R, implies that the left path from X_1 is shorter than the right path from X_1. Hence the direction of motion at X_1 remains toward V_L. In the second case, the robot moves from X_0 to X_1 toward B_L, such that the other endpoint, B_R, is an occlusion node that slides along an obstacle edge. As illustrated in Figure 7b, the length of the left path through B_L only decreases during this motion, while the length of the right path through B_R only increases. Hence the left path remains shorter than the right path, and the robot retains its direction toward B_L. □

We are now ready to show that type 2 events cause finitely many direction changes. For a given obstacle edge e, let the *extended edge* of e be the line segment that extends from e into the free space.

LEMMA 6. During motion toward the target, there are finitely many successive direction changes due to type 2 events.

Proof Sketch. In the proof (which appears in the appendix), we note that a node is removed or added to the LTG (a possible type 2 event) in one of the following two ways. When the robot crosses an extended edge, both tangent nodes and occlusion nodes can change. When the robot crosses a *bitangent edge*, which is the line segment in the free space tangent to two convex obstacle vertices, only occlusion nodes can change. First, we show that there is no change of direction when the robot crosses a bitangent edge. Then we show that if the crossing of an extended edge causes a direction change, the new direction must be along the extended edge the robot has just crossed. The robot then traces the extended edge until it crosses another extended edge, or until an event of some other type occurs. Since the extended edges form a fixed arrangement of lines, the number of successive direction changes due to type 2 events is finite. □

The next lemma discusses the type 3 events, where the edge containing the focus node ceases to be admissible.

LEMMA 7. During motion toward the target, there are finitely many type 3 events.

Proof Sketch. Let a *blocking extended edge* be an extended edge that starts at a convex vertex of the blocking obstacle. The blocking extended edges partition the free space into cells. In the appendix, we first consider the case where the robot is moving in the interior of a cell, illustrated in Figure 8a. Suppose a type 3 event occurs at the point X_0 while the robot is moving toward B_L^{x0}. Now the robot continues toward B_R^{x0} and reaches a point X_1, which still lies in the interior of the cell. We show that if a second type 3 event occurs at X_1, all the LTG nodes V at X_1 satisfy $d(V, T) > d(X_1, T)$. This condition implies that the robot must have switched to boundary-following at some earlier point. Hence, at most, one type 3 event can occur in the interior of a cell. We then consider the case illustrated in Figure 8b, where a type 3 event occurs while the robot is moving along a blocking extended edge. As the figure shows, after the type 3 event occurs, the robot's new direction is toward the other endpoint of the blocking obstacle, and the obstacle edge e becomes *invisible*. Thus each time the robot moves away from a blocking extended edge, some obstacle edge e becomes invisible. Furthermore, until the robot reaches one of the endpoints of the blocking obstacle (a type 1 event), an obstacle edge that has become invisible cannot become visible again. Thus the number of type 3 events until a type 1 event occurs is finite. Since according to Lemma 3 the number of type 1 events is finite, the total number of type 3 events is also finite. □

The following proposition characterizes the path of TangentBug during motion toward the target.

PROPOSITION 2. Using unlimited sensor range, each motion-toward-the-target segment has finite length.

Proof. TangentBug generates a piecewise-linear path. Hence, it suffices to show that the path consists of finitely many linear segments of finite length. Each linear segment corresponds to a particular direction of motion. According to Lemmas 3 and 7, the number of direction changes due to type 1 and type 3 events is finite. According to Lemma 6, there are finitely many successive type 2 events. Type 2 events are succeeded either by type 1 or type 3 events, or by termination of the motion toward the target. Hence there are finitely many direction changes due to type 2 events in each motion-toward-the-target segment. Thus there are finitely many linear segments in each motion-toward-the-target segment. The length

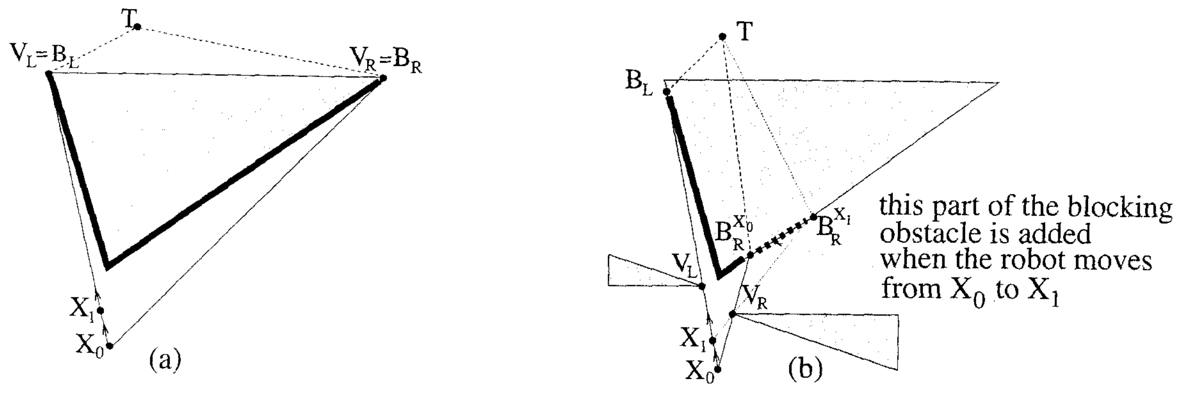
T
V_L=B_L
V_R=B_R
X_1
X_0
(a)
T
B_L
B_R^{X_0}
B_R^{X_1}
V_L
V_R
X_1
X_0
(b)
this part of the blocking obstacle is added when the robot moves from X_0 to X_1

Fig. 7. There are no type 2 events (a) when the LTG nodes remain fixed, and (b) when an occlusion node slides along an obstacle edge.

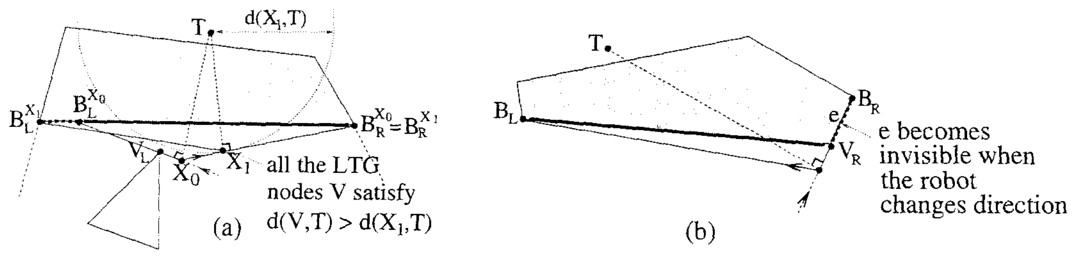
T
d(X_1,T)
B_L^{X_1}
B_L^{X_0}
B_R^{X_0}=B_R^{X_1}
V_L
X_0
X_1
all the LTG nodes V satisfy d(V,T) > d(X_1,T)
(a)
T
B_L
B_R
e
V_R
e becomes invisible when the robot changes direction
(b)

Fig. 8. A type 3 event (a) while moving between blocking extended edges; and (b) while moving along a blocking extended edge.

of each linear segment is finite, since the robot moves toward a fixed focus node. □

The next proposition characterizes the behavior of TangentBug during the boundary-following mode of motion.

PROPOSITION 3. Using unlimited sensor range, each boundary-following segment has finite length. Moreover, if the target T is reachable from the point where the robot switches to boundary-following, the leaving condition will cause the robot to leave the obstacle.

Proof Sketch. Let $\mathcal{O}$ denote the boundary of the obstacle being followed. By definition of TangentBug, the robot goes around $\mathcal{O}$ at most once before it terminates its motion. In Kamon, Rimon, and Rivlin (1995), we show that the length of this segment is bounded by the perimeter of the obstacle (plus a term that is bounded by the distance $\|S-T\|$). In the appendix, we focus on showing that the leaving condition, $d(V,T) < d_{min}(T)$ for some $V \in$ LTG, is satisfied during the boundary-following. First we consider the case where T is visible from some point $Q \in \mathcal{O}$. Since the robot follows $\mathcal{O}$ in a fixed direction, it must cross the line segment $[Q,T]$, and at the crossing, T becomes a node of the LTG. Therefore the leaving condition will hold in this case. Next we consider the case where T is not directly visible from any point of $\mathcal{O}$. Let C be a point on $\mathcal{O}$ which is closest to T. Let C' be the point on the neighboring obstacles which is the closest to C along the line segment $[C,T]$. During the boundary-following, the minimal distance to T satisfies $d_{min}(T) \geq d(C,T)$. Since T is reachable from $\mathcal{O}$, it must be possible to move from C directly toward T. Hence $d(C',T) < d(C,T)$, and subsequently, $d(C',T) < d_{min}(T)$. The leaving condition is therefore satisfied when C' becomes a node of the LTG. The remainder of the proof consists of showing that C' must become a node of the LTG during the boundary-following. □

We have shown that both modes of motion terminate after a finite-length path. Next, we show that there are finitely many segments of these motion modes. The following lemma and its corollary give relevant properties of the condition for switching between the two modes.

LEMMA 8. Every switch point P_i, where the robot switches from motion toward the target to boundary-following, has a corresponding local-minimum point M_i of the distance function $d(w,T)$, such that $d(M_i,T) \leq d(P_i,T)$.

Proof. At the switch point P_i, all the nodes V of the subgraph G_{I} satisfy $d(V,T) > d(P_i,T)$. Since the robot switches to boundary-following at P_i, there must be a blocking obstacle between P_i and T. Let Q be the point where the line segment $[P_i, T]$ intersects the blocking obstacle. Clearly $d(Q,T) \leq d(P_i,T)$. Since the nodes V of G_1 satisfy $d(V,T) > d(P_i,T)$, the endpoints of the blocking obstacle satisfy $d(B_L,T) > d(P_i,T)$, and $d(B_R,T) > d(P_i,T)$. Therefore, $d(B_L,T) > d(Q,T)$, and $d(B_R,T) > d(Q,T)$. Since the blocking obstacle is a continuous curve whose endpoints are B_L and B_R, there must be a local-minimum M_i of $d(w,T)$ somewhere on the blocking obstacle. Since the point Q satisfies $d(Q,T) \leq d(P_i,T)$, the local minimum that is the closest to T satisfies $d(M_i,T) \leq d(P_i,T)$. □

COROLLARY 1. Let M_i be the local minimum of $d(w,T)$ associated with the switch point P_i. Then the distance to the target decreases between successive local-minimum points, that is, $d(M_{i+1},T) < d(M_i,T)$.

Proof. Starting at P_i, the robot follows an obstacle boundary, leaves it, and then performs a transition phase followed by motion toward the target until it reaches the next switch point P_{i+1}. During the initial boundary-following, $d_{min}(T) \leq d(M_i,T)$. The leaving condition holds when an LTG node, V_{leave}, satisfies $d(V_{leave},T) < d_{min}(T)$. In the transition phase, the robot moves toward V_{leave} until it reaches a point Z_i that satisfies $d(Z_i,T) < d_{min}(T)$. The motion toward the target is resumed at Z_i. According to Lemma 1, the distance $d(x,T)$ decreases during motion toward the target. Hence $d(P_{i+1},T) \leq d(Z_i,T)$. According to Lemma 8, $d(M_{i+1},T) \leq d(P_{i+1},T)$. Thus $d(M_{i+1},T) \leq d(P_{i+1},T) \leq d(Z_i,T) < d_{min}(T) \leq d(M_i,T)$, and the result follows. □

The following theorem asserts that TangentBug always terminates.

THEOREM 1. Using unlimited sensor range, TangentBug always terminates after following a finite-length path.

Proof. First, we show that there are finitely many segments of each motion mode. According to Lemma 8, every switch point P_i is associated with a local-minimum point M_i of $d(w,T)$. According to Corollary 1, $d(M_i,T)$ decreases at successive local-minimum points. Thus each point M_i is associated with at most one switch to boundary-following. Since the number of local minima of $d(w,T)$ is finite, the path consists of finitely many boundary-following segments. Since two consecutive boundary-following segments are interleaved by a single motion-toward-the-target segment and a single transition segment, there are also finitely many motion-toward-the-target and transition segments. Next, consider the length of each motion segment. Proposition 2 guarantees that the path length of each motion-toward-the-target segment is finite. The path length of each transition phase is finite because the robot moves toward a fixed focus point. Proposition 3 guarantees that the path length of each boundary-following segment is finite. Hence TangentBug always terminates after a finite-length path. □

The following theorem asserts that TangentBug is complete.

THEOREM 2. Using unlimited sensor range, TangentBug always finds the target if it is reachable from the start point.

Proof. As discussed in the proof of Theorem 1, there are finitely many boundary-following segments. If T is reachable from the start point, Proposition 3 guarantees that the leaving condition will cause the robot to terminate each boundary-following segment and leave the obstacle boundary. Since every such segment is followed by a transition phase, there is a *last* transition phase. The last transition phase either terminates at T, or is followed by the last motion-toward-the-target segment, which terminates at T. □

Finally, we discuss an upper bound on the path length of TangentBug, making the following two assumptions. Given a vector v, let $\hat{v} = v/\|v\|$. The nodes V of the subgraph G_1 satisfy $\widehat{(V-x)}\cdot\widehat{(T-x)} > 0$, where x is the current robot location. We assume that the subgraph G_1 is further restricted to the nodes V that satisfy $\widehat{(V-x)}\cdot\widehat{(T-x)} > \gamma$, where γ is a small positive parameter. The second assumption is concerned with the leaving condition. The robot is allowed to seek candidate nodes which satisfy the leaving condition only within a range of ρ, where ρ is a positive and finite parameter.

PROPOSITION 4. Using unlimited sensor range and under the given assumptions, an upper bound L_{max} on the path length of TangentBug is

$$L_{max} = \left(1 + \frac{1}{\gamma}\right)\|S-T\| + \sum_{i \in \mathcal{I}} \sharp\,\mathrm{Minima}_i \times (\Pi_i + \rho), \tag{1}$$

where $\mathcal{I}$ is the index set of the obstacles that intersect the disc D_T, Π_i is the perimeter of the i^{th} obstacle in $\mathcal{I}$, and $\sharp\,\mathrm{Minima}_i$ is the number of local minima of $d(w,T)$ in D_T along the ith obstacle boundary. (D_T is the disc of radius $\|S-T\|$ centered at T.)

Proof Sketch. First, consider the term $\frac{1}{\gamma}\|S-T\|$. The robot moves only along admissible LTG edges, whose angle α with respect to the direction toward T satisfies $\cos\alpha > \gamma$. Hence the ratio between the decrement in $d(x,T)$ and the distance traveled along the path is bounded by γ. Since $d(x,T)$ decreases monotonically during the robot's motion, $\frac{1}{\gamma}\|S-T\|$ bounds the total length of the motion toward the target segments. Next, consider the term $\|S-T\| + \sum_{i\in\mathcal{I}} \sharp\,\mathrm{Minima}_i \times \Pi_i$. We showed in Proposition 1 that the number of boundary-following segments is bounded by $\sum_{i\in\mathcal{I}} \sharp\,\mathrm{Minima}_i$. As an intermediate step, consider the path length that would be generated if the robot turns its range sensor off at the switch points, and performs the boundary-following using a contact sensor. In this case, every boundary-following segment consists of a subsegment where the robot moves from a switch point P_i toward T until it hits an obstacle at a point H_i, and a subsegment where the robot follows the obstacle boundary. The length of each boundary-following subsegment is bounded by the obstacle's perimeter. The accumulated sum of the $[P_i, H_i]$ subsegments is bounded by $\|S-T\|$. When a range sensor is used during the boundary-following, the resulting path can only be shorter than the path generated with a contact sensor. Thus $\|S-T\| + \sum_{i\in\mathcal{I}} \sharp\,\text{Minima}_i \times \Pi_i$ bounds the total length of boundary-following segments. Last, the length of each transition phase is bounded by ρ, and the term $\sum_{i\in\mathcal{I}} \sharp\,\text{Minima}_i \times \rho$ bounds the total length of the transition phases. □

The reader should note that eq. 1 is only a worst-case bound on the performance of TangentBug. The average performance of TangentBug is characterized in the next section.

5. Experimental Results

The experimental study of TangentBug consists of simulations and experiments on a mobile robot. The simulations study the dependence of TangentBug's paths on the sensor range R, and compare TangentBug with the classical VisBug algorithm. The experiments demonstrate the practical usefulness of the algorithm, and we discuss several details of the implementation.

5.1. Simulation Results

We tested TangentBug on the following three classes of environments. The first class consisted of disjoint convex obstacles (Fig. 9); the second class consisted of mazelike obstacles (Fig. 10); and the third class consisted of obstacles with "officelike" shapes (Fig. 11). Nine environments were constructed for each class, and 100 randomly chosen start/target points were used in each environment, giving a total number of 2,700 different paths. Furthermore, five maximal detection range values, varying from 0 to ∞, were tested on each of the 2,700 paths. More information about the simulated environments is provided in Kamon, Rimon, and Rivlin (1995).

For comparison, we also tested the classical VisBug algorithm on the same sample environments. We have implemented the VisBug21 version of VisBug, described in Lumelsky and Skewis (1990). This algorithm uses the range data to make local shortcuts relative to the path that would be planned by the contact-sensor algorithm Bug2 (Lumelsky and Stepanov 1987). Under Bug2, the robot moves directly toward the target until it hits an obstacle. Then it follows the obstacle boundary using a predefined direction (clockwise), until a leaving condition holds. The leaving condition is tested only on the line $[S, T]$, and it guarantees that the distance between the hit points and T decreases monotonically. We have also tested a modified version of VisBug, in

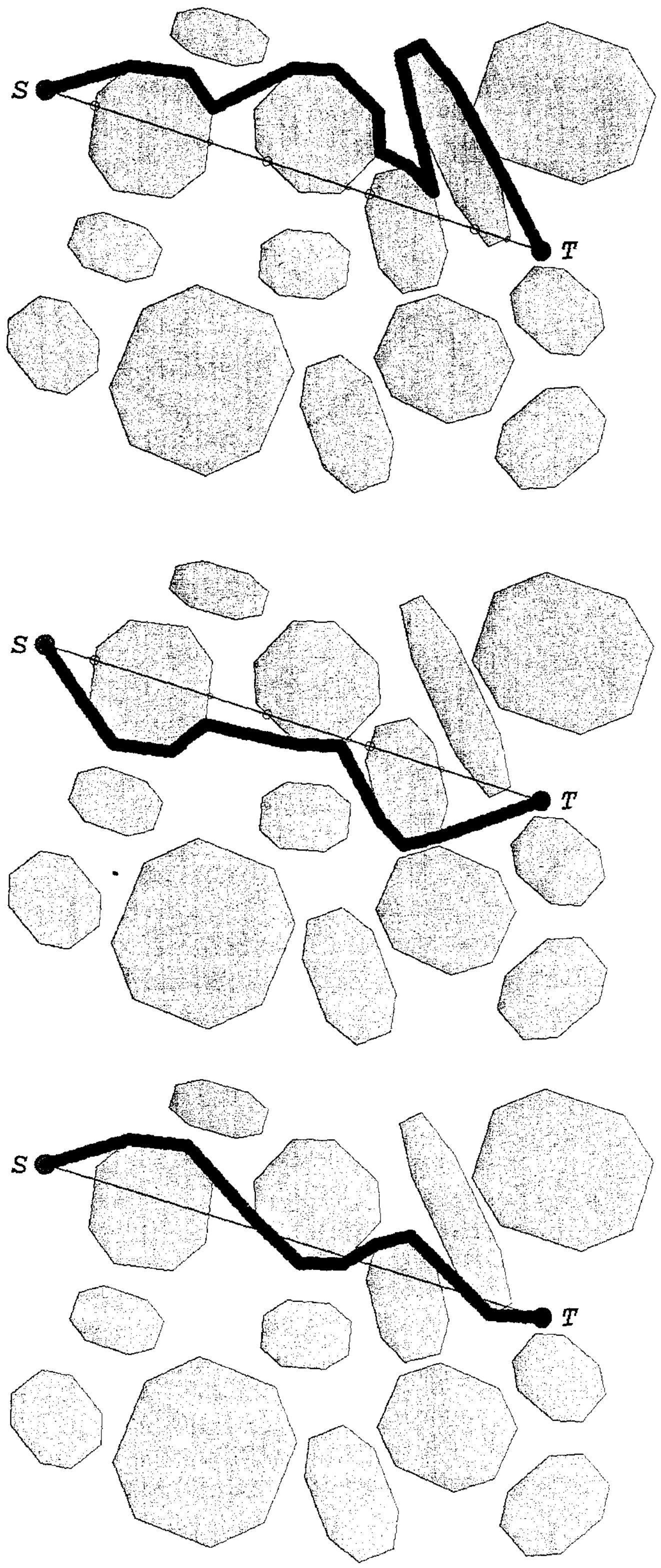

Fig. 9. Simulation results using unlimited sensor range among convex obstacles: (a) VisBug (path length 1.52); (b) VisBug with local turning (path length 1.06); and (c) TangentBug (path length 1.00).

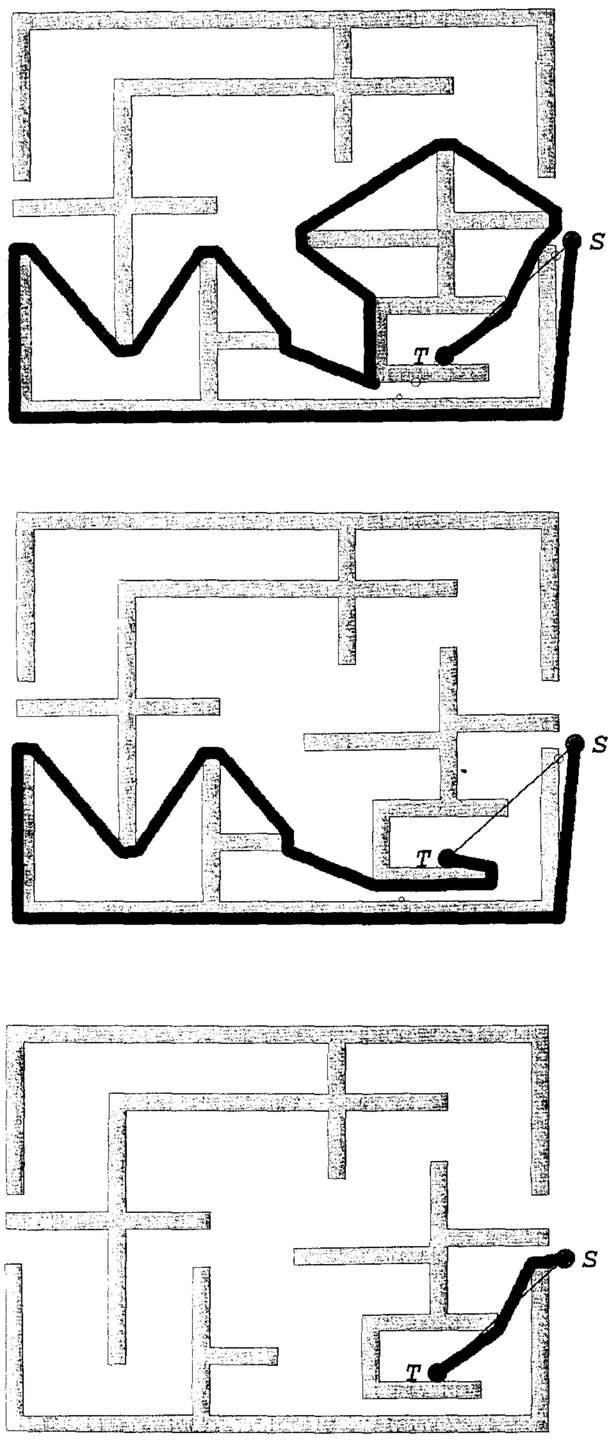

Fig. 10. Simulation results using unlimited sensor range in a maze: (a) VisBug (path length 11.11); (b) VisBug with local turning (path length 8.44); and (c) TangentBug (path length 1.00).

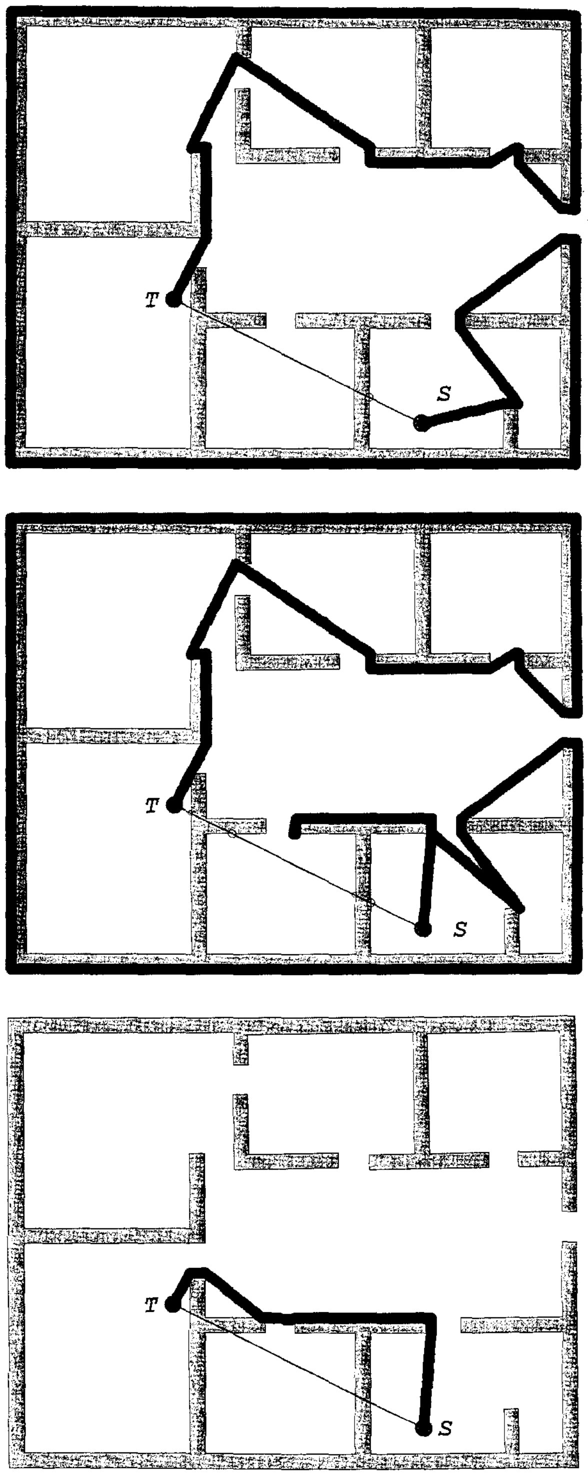

Fig. 11. Simulation results using unlimited sensor range in an officelike environment: (a) VisBug (path length 7.92); (b) VisBug with local turning (path length 9.11); and (c) TangentBug (path length 1.03).

which the boundary-following direction is chosen as follows. When the robot hits an obstacle, it calculates a segment of length ϵ tangent to the obstacle boundary at the hit point. If the left endpoint of the tangent segment is closer to T than the right endpoint, the robot turns left and follows the boundary in a clockwise direction. Otherwise, the robot turns right and follows the boundary in a counterclockwise direction.

The results of running the three algorithms on the three classes of environments are summarized in Tables 1, 2, and 3. The paths produced by TangentBug were shorter than the paths produced by VisBug in all tested scenarios. Increasing the sensor detection range, R, improved the performance of the three algorithms. But TangentBug produced shorter paths for all values of R. Each table presents the average path length, measured over 900 runs for the specified detection range, relative to the globally shortest path in the environment. We also present the ratio (in percentages) between the average path lengths of TangentBug and VisBug. This ratio indicates an average improvement factor of three for unlimited sensor range in the officelike environments.

The simulations also provide us with the following insight into the dependence of the path length on the sensor detection range R. In the simple environments consisting of convex obstacles, the paths produced by TangentBug using unlimited sensor range resemble the globally optimal paths. In our experiments, TangentBug's paths were identical to the globally optimal ones in 69% of the runs, and their average path length was 1.03 relative to the optimal path length. In these simple environments, the algorithm used mostly the motion-toward-the-target mode, implying that the range data was continuously used for choosing the locally optimal direction. The minor improvement in the path length as the sensor range increases suggests that in simple environments small sensor range is sufficient, since in such environments the local data usually leads to the globally correct decisions. In the more complex environments, the three algorithms achieved significant local shortcuts by scanning the boundaries of concave obstacles instead of actually following them; hence the effect of increasing the sensor detection range was more pronounced in the complex environments.

Note that TangentBug uses the range data more effectively than the other algorithms. For example, Figure 12 shows the paths generated by TangentBug in an officelike environment for increasing values of R. The advantage of TangentBug is more apparent in the officelike environments for two reasons: first, these environments contain obstacles with long perimeters, which cause long paths when the boundary-following direction is not the globally correct one. TangentBug produces shorter paths in these environments by circumventing obstacles using the motion-toward-the-target mode, without committing the robot to a fixed boundary-following direction. TangentBug also chooses the boundary-following direction based on all the local information, and it can leave an obstacle boundary much before the line $[S, T]$ is visible. The second reason is that officelike environments contain some built-in regularity, which makes the local information more relevant to the globally correct decisions.

Finally, we note that the variation in the performance of TangentBug among the sample environments is quite small. Table 4 shows the minimal, average, and maximal ratios between the average path-lengths of TangentBug and VisBug for each class of environments, using unlimited sensor range and considering all nine sample environments of each class. The minimal and maximal values are the ratios between average path lengths of 100 simulated runs of TangentBug and VisBug in specific environments.

Table 1. TangentBug Performance among Convex Obstacles, Compared to VisBug[a]

R	VisBug	VisBug + Turning	Tangent-Bug	$\frac{\text{TangentBug}}{\text{VisBug}}$
0	1.56	1.17	1.09	70%
50	1.36	1.09	1.04	76%
100	1.31	1.07	1.03	79%
200	1.29	1.06	1.03	80%
∞	1.28	1.06	1.03	80%

a. Five detection ranges are tested in environments whose width and height are 800 × 700 units. The average path length is expressed relative to the globally shortest path. The left column presents the ratio (in percentages) between the average path lengths of TangentBug and VisBug.

Table 2. TangentBug Performance in Mazelike Environments, Compared to VisBug

R	VisBug	VisBug + Turning	Tangent-Bug	$\frac{\text{TangentBug}}{\text{VisBug}}$
0	3.82	3.63	3.43	90%
50	3.32	3.17	2.82	85%
100	2.71	2.59	2.02	75%
200	2.21	2.10	1.45	66%
∞	2.10	2.02	1.34	64%

Table 3. TangentBug Performance in Officelike Environments, Compared to VisBug

R	VisBug	VisBug + Turning	Tangent-Bug	$\frac{\text{TangentBug}}{\text{VisBug}}$
0	9.86	8.38	7.10	72%
50	8.78	7.40	5.52	63%
100	7.28	6.11	3.64	50%
200	5.22	4.34	1.48	28%
∞	4.14	3.36	1.38	33%

Table 4. Minimal, Average, and Maximal Ratios between Average Path Lengths of TangentBug and VisBug Using Unlimited Sensor Range, Computed over the Sample Environments in Each Class (in percentages)

Class	Minimal Ratio	Average Ratio	Maximal Ratio
Convex obstacles	75	80	83
Maze obstacles	56	64	72
Office obstacles	25	33	38

5.2. Experiments in a Real-World Scenario

To demonstrate the practical usefulness of TangentBug, we have implemented and tested the algorithm on a Nomad 200 mobile robot. The robot was equipped with a ring of 16 sonar sensors and a ring of 16 infrared range sensors that were used for collecting the range data. In the experiments, no model of the environment was supplied to the robot, and its decisions were based solely on the range data read by the sensors. To adjust the algorithm to the available sensory information, we made the following slight modifications. During motion toward the target, only the LTG edges that pointed toward T_{node} or were tangent to the blocking obstacle were explicitly constructed. (The algorithm is still complete under this restriction.) Further, given the low (22.5°) angular resolution of the sensors, we estimated the direction of the blocking-obstacle endpoints as follows. When no blocking obstacle was detected, T_{node} was placed along the direction toward T. When a blocking obstacle was detected, the right (left) edge was chosen as the first sensor direction, which indicated that there was no obstacle to the right (left) side of the blocking obstacle.

We performed many experiments with the robot, and one such experiment is shown in Figure 13. In this experiment the distance from the start to the target was 5.7 m, and the robot had to maneuver around two obstacles that blocked its way to the target. The path is shown in Figure 13b as a bold line, and the sensed obstacles are rendered as an accumulation of sonar readings along the path. Although the sonar measurements contained substantial noise, they were mostly used for locating the endpoints of the blocking obstacle, not for exact distance measurements. The following data was averaged over five successive runs in this experimental setting. The average path length was 6.1 m. The average run time was 52 sec, giving an average speed of 11.5 cm/sec. We also measured the time it took the robot to perform a basic motion step of the algorithm. This time was 175 msec during motion toward the target, and 390 msec during the boundary-following.

6. Concluding Discussion

We presented TangentBug, a range-sensor-based globally convergent navigation algorithm for 2-DOF mobile robots. The algorithm expands on the existing Bug-family algorithms in two respects. First, TangentBug is specifically designed for using range data. Second, TangentBug incorporates the notion of the locally shortest path into the general Bug paradigm. We introduced a local range-data-based version of the classical tangent graph, termed the local tangent graph or LTG, and reformulated the two basic behaviors of the Bug family in a way that continuously uses the LTG. We also presented a new convergence mechanism that was based on local minima of the distance from the target along obstacle boundaries, and defined new transition conditions that implemented this convergence mechanism.

The simulations show substantial improvement in the average path length as the sensor's detection range increases; for instance, we observed an improvement from 7.10 to 1.38 in officelike environments. The simulations also indicated a significant advantage of TangentBug relative to the classical VisBug algorithm. For example, in officelike environments the average path length is 33% relative to VisBug, using infinite sensor range. We have also implemented TangentBug on a mobile robot, demonstrating the usefulness of the algorithm. TangentBug is especially simple to implement, since it uses purely reactive decisions that are based on the currently visible obstacles. Moreover, the globally convergent behavior is integrated into the algorithm in a way that minimizes the need for accurate global positioning, which is difficult to obtain with today's sensors.

Finally, we note that TangentBug provides a framework for various extensions. For example, the algorithm can be adapted for navigating among piecewise-smooth obstacles, and can be adjusted for practical use in dynamic environments. Additional information can be easily incorporated to improve performance in practical scenarios. For example, by modifying the expected path length from the LTG nodes to the target, based on a nominal global model of the environment or accumulated sensory data, we expect even better performance. Such extensions are currently under investigation.

Appendix: Proof Details

The following lemma characterizes the ways in which type 2 events can occur. The length of a virtual edge from a node V of G_1 to T is denoted $e\_len_x(V)$.

LEMMA 9. Type 2 events occur only when an LTG node, which is either a tangent or an occlusion node, is removed or added to the LTG.

Jsing

he no-
adigm.
e clas-
r LTG,
family
esented
al min-
ndaries,
ted this

the av-
creases;
to 1.38
icated a
lassical
nments
sing in-
entBug
ie algo-
since it
urrently
behav-
nimizes
ficult to

nework
can be
stacles,
nviron-
rated to
nple, by
es to the
onment
perfor-
ition.

1 type 2
node V

3 node,
oved or

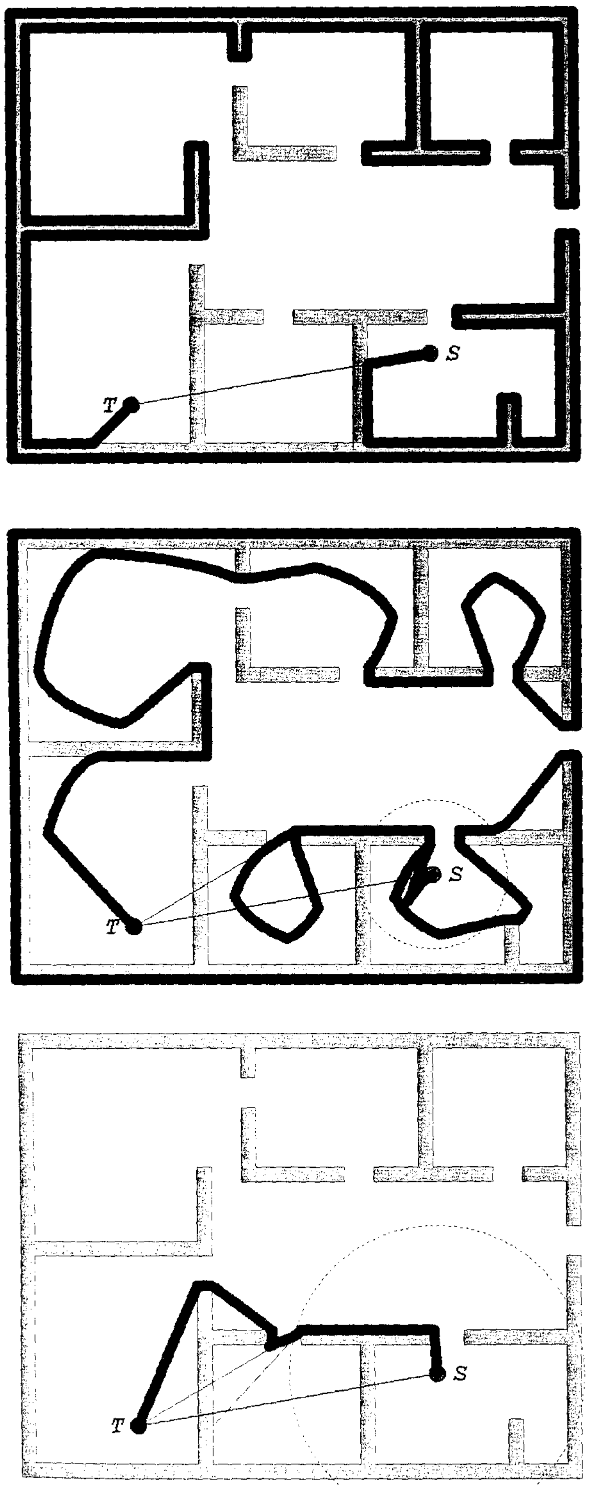

Fig. 12. The effect of increasing the maximal detection range in an officelike environment whose size is 800 × 700 units: (a) TangentBug with a contact sensor (path length 12.0); (b) TangentBug with a detection range of 100 units (path length 10.92); and (c) TangentBug with a detection range of 200 units (path length 1.08).

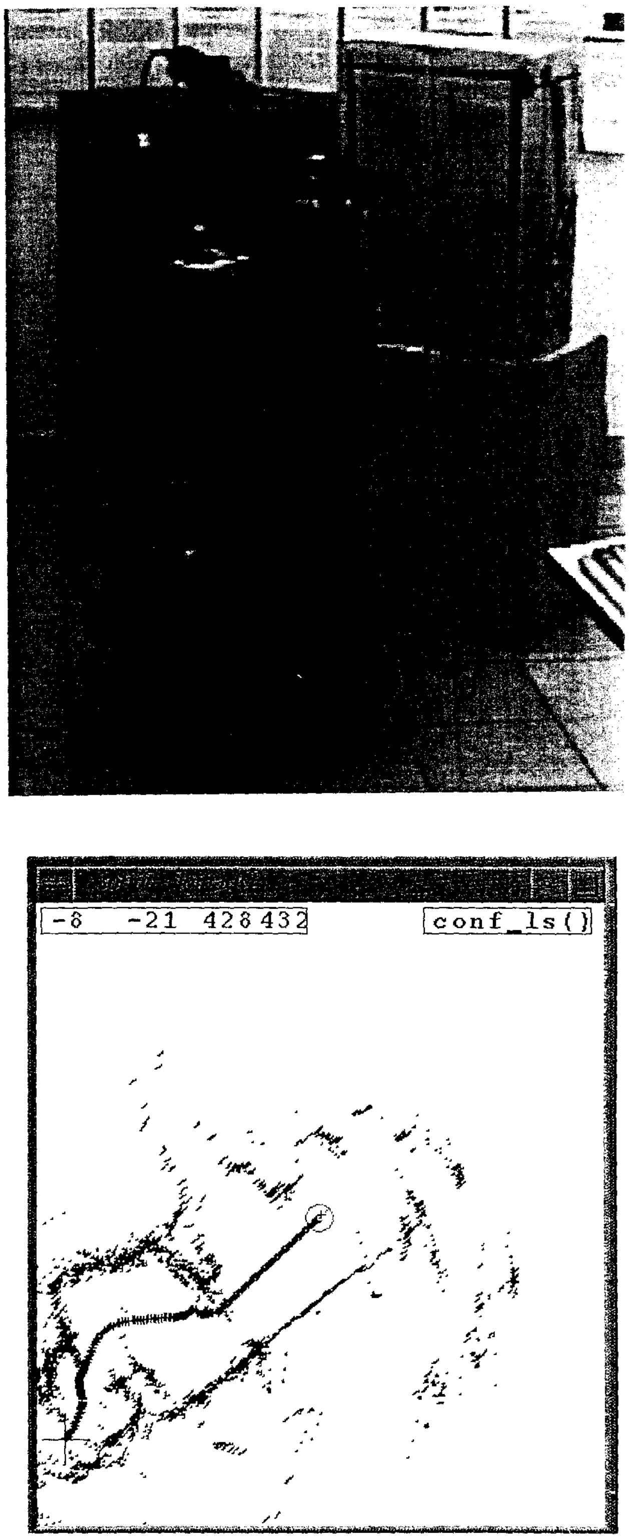

Fig. 13. Top: the Nomad robot. Bottom: the trace of a single run, shown on the robot's graphical user interface. The start point is marked with a cross, and the target with a circle.

Proof. Consider a motion of the robot such that no node is removed or added to the LTG. In this case, the LTG nodes either remain fixed, or some occlusion nodes slide along obstacle edges. First, suppose that the robot moves from X_0 to X_1 in a way that keeps all the LTG nodes fixed. Without loss of generality, assume that the shortest path from X_0 to T on the augmented graph is the left path. Since the left path is shorter than the right path, we have that

$$\|X_0 - V_L\| + e\_len_{X_0}(V_L) < \|X_0 - V_R\| + e\_len_{X_0}(V_R). \tag{2}$$

Since all the LTG nodes remain fixed as the robot moves from X_0 to X_1, the left and right paths from X_1 pass through the same nodes V_L and V_R. Moreover, the lengths of the expected paths from V_L and V_R to T do not change as the robot moves from X_0 to X_1. Substituting $e\_len_{X_0}(V_L) = e\_len_{X_1}(V_L)$ and $e\_len_{X_0}(V_R) = e\_len_{X_1}(V_R)$ in eq. (2) gives

$$\|X_0 - V_L\| + e\_len_{X_1}(V_L) < \|X_0 - V_R\| + e\_len_{X_1}(V_R). \tag{3}$$

The term $\|X_0 - V_L\|$ on the left side can be written as $\|X_0 - V_L\| = \|X_1 - V_L\| + \|X_1 - X_0\|$, since the points X_0, X_1, and V_L lie on the same straight line. Hence eq. (3) can be written as $\|X_1 - V_L\| + e\_len_{X_1}(V_L) < \|X_0 - V_R\| - \|X_1 - X_0\| + e\_len_{X_1}(V_R)$. But $\|X_0 - V_R\| < \|X_0 - X_1\| + \|X_1 - V_R\|$ according to the triangle inequality. Hence

$$\|X_1 - V_L\| + e\_len_{X_1}(V_L) < \|X_1 - V_R\| + e\_len_{X_1}(V_R).$$

Thus the left path from X_1 is shorter than the right path from X_1, and the direction of motion remains toward V_L (Fig. 7a).

Now we show that no event of type 2 occurs when an occlusion node slides continuously along an obstacle edge. Without loss of generality, assume that the shortest path from X_0 to T on the augmented graph is the left path. Suppose that B_R is an occlusion node which slides on an edge of the blocking obstacle. As illustrated in Figure 7b, we denote by B_R^{x0} and B_R^{x1} the point B_R as seen from X_0 and X_1. The part of the blocking-obstacle boundary that is visible from X_1, (B_L, B_R^{x1}), includes all the visible boundary from X_0, (B_L, B_R^{x0}), and extends to the right side of (B_L, B_R^{x0}). Therefore the path length from V_R to T via B_R^{x1} is longer than via B_R^{x0}. (The same argument holds for more complex situations, where the shortest path from B_R^{x1} to T is not the straight line $[B_R^{x1}, T]$.) Hence the length of the shortest path from B_R to T increases when the robot moves from X_0 to X_1, $e\_len_{X_1}(V_R) > e\_len_{X_0}(V_R)$. Based on Lemma 4, the left path leads directly toward B_L. Therefore $e\_len_{X_1}(V_L) = e\_len_{X_0}(V_L)$. Given that eq. (2) holds at X_0, it follows that eq. (3) holds true. Hence, using an argument similar to the one used above, the left path remains shorter than the right path at X_1, and the robot retains its direction toward B_L. □

The next lemma asserts that type 2 events cause finitely many direction changes.

LEMMA 10. During motion toward the target, there are finitely many successive direction changes due to type 2 events.

Proof. According to Lemma 9, type 2 events occur only when an LTG node is removed or added to the LTG. Such a change in the LTG can occur in one of the following two ways. When the robot crosses an extended edge, both tangent nodes and occlusion nodes can change. When the robot crosses a bitangent edge, which is the line segment in the free space tangent to two convex obstacle vertices, only occlusion nodes can change.

First, consider the case where the robot crosses a bitangent edge. According to Lemma 4, the candidate shortest paths always lead directly to the endpoints of the blocking obstacle. Hence the direction of motion can change only when the occlusion node which is removed or added to the LTG is one of the endpoints of the blocking obstacle. If an occlusion node appears on the blocking obstacle, it must be that just before the crossing some obstacle was lying between the robot and the blocking obstacle. But this contradicts the definition of the blocking obstacle, which is visible in its entirety from the current robot location. Hence the crossing of a bitangent edge cannot cause the addition of an occlusion node to the blocking obstacle. The crossing also cannot cause the removal of an occlusion node. Rather, the crossing transforms an occlusion node into a tangent node in the LTG. Since the location of the node does not change in this event, there is no change in the shortest path to T, and the robot continues in the same direction of motion. Thus there is no change of direction when the robot crosses a bitangent edge.

Consider now the crossing of an extended edge. The robot always moves toward an endpoint of the blocking obstacle. Suppose that while moving toward the left endpoint B_L, the robot crosses an extended edge. Suppose, too, that the crossing causes a change in an LTG node which triggers a change of direction. Using Lemma 4 again, the candidate shortest paths always lead directly to the endpoints of the blocking obstacle. Thus, only a change in B_L or B_R can possibly cause a change of direction. Since the robot was moving toward B_L, the other endpoint, B_R, must have changed when the robot crossed the extended edge. Furthermore, the change in the LTG nodes must have occurred along the extended edge. Hence, immediately after the crossing, the robot must be moving toward B_R along the extended edge.[4] The robot then traces the extended edge until it crosses another extended edge, or until an event of some other type occurs. Since the extended edges form a fixed arrangement of lines with a finite number of intersection points, the number of successive direction changes due to type 2 events is finite. □

4. More precisely, after the robot crosses the extended edge, it moves toward B_R in a motion that resembles a tracing of the extended edge.

The next lemma discusses the type 3 events.

LEMMA 11. During motion toward the target, there are finitely many type 3 events.

Proof. Let a *blocking extended edge* be an extended edge that starts at a convex obstacle vertex of the blocking obstacle. The blocking extended edges partition the free space into cells. We consider separately the case where the robot is moving in the interior of a cell, and the case where it is moving along a blocking extended edge. For simplicity, we ignore the nongeneric case where the robot is crossing a blocking extended edge such that a type 3 event occurs at the crossing point.

First, consider the motion in the interior of a cell, illustrated in Figure 8a. We wish to show that at most one type 3 event occurs in the interior of a cell during each motion-toward-the-target segment. Suppose a type 3 event occurs at the point X_0 while the robot is moving toward the node $B_L^{x_0}$. Since the edge $[X_0, B_L^{x_0}]$ ceases to be admissible, X_0 is a minimum of the distance function $d(w, T)$ along $[X_0, B_L^{x_0}]$. In particular, we have that $d(B_L^{x_0}, T) > d(X_0, T)$. Since the angle α between the edge $[X_0, B_L^{x_0}]$ and the line segment $[X_0, T]$ is 90°, all the LTG edges to the left of $[X_0, B_L^{x_0}]$ also become nonadmissible. Thus all the LTG nodes V to the left of the line segment $[X_0, T]$ satisfy $d(V, T) > d(X_0, T)$. Now the robot continues toward the right endpoint $B_R^{x_0}$ and reaches a point X_1, which still lies in the interior of the cell. Since the distance to T decreases during motion toward the target, $d(X_1, T) < d(X_0, T)$.

Assume by contradiction that a type 3 event also takes place at X_1. Using an argument similar to the one given above, all the LTG nodes V to the right of $[X_1, T]$ satisfy $d(V, T) > d(X_1, T)$. Since the robot has not crossed any blocking extended edge, the region in the free space bounded by the edges $[X_0, B_L^{x_0}]$, $[X_0, B_R^{x_0}]$, and the blocking obstacle are also visible from X_1 (Fig. 8a). Moreover, there are no LTG nodes in the interior of this region. This information, together with the inequality $d(X_1, T) < d(X_0, T)$, implies that *all* the LTG nodes V at X_1 satisfy $d(V, T) > d(X_1, T)$. But this is the condition that terminates the motion-toward-the-target mode and initiates the boundary-following mode. Hence the robot must have switched to boundary-following mode at some earlier point. Thus, at most one type 3 event can occur in the interior of a cell.

Consider now the case where a type 3 event occurs while the robot is moving along a blocking extended edge. We already know that the robot must be moving toward one of the endpoints of the blocking obstacle. Suppose it is moving along a blocking extended edge toward the right endpoint B_R, as illustrated in Figure 8b. By definition of a blocking extended edge, there exists an edge e of the blocking obstacle along the blocking extended edge. After the type 3 event occurs, the robot's new direction must be toward the left endpoint B_L. Since the robot leaves the blocking extended edge and moves toward B_L, the obstacle edge e must become *invisible*, as shown in Figure 8b. Thus, each time the robot moves away from a blocking extended edge, some obstacle edge e becomes invisible. Furthermore, until the robot reaches one of the endpoints of the blocking obstacle (a type 1 event), an obstacle edge that has become invisible cannot become visible again.

We can therefore summarize the two cases and say that the number of type 3 events until a type 1 event occurs is finite. Since according to Lemma 3 the number of type 1 events is finite, the total number of type 3 events is also finite. □

The next proposition characterizes the behavior of TangentBug during the boundary-following mode of motion.

PROPOSITION 5. Using unlimited sensor range, each boundary-following segment has finite length. Moreover, if the target T is reachable from the point P_i where the robot switches to boundary-following, the leaving condition will cause the robot to leave the obstacle.

Proof. Let $\mathcal{O}$ denote the boundary of the obstacle being followed. (The obstacle can have several boundary components, and $\mathcal{O}$ is the component visible to the robot.) By definition of TangentBug, the robot goes around $\mathcal{O}$ at most once before it terminates its motion. In Kamon, Rimon, and Rivlin (1995), we show that the length of this segment is bounded by the perimeter of the obstacle, plus a term that is bounded by the distance $\|S - T\|$. We now focus on showing that the leaving condition, $d(V, T) < d_{min}(T)$ for some $V \in$ LTG, is satisfied during the boundary-following. First consider the case where T is visible from some point $Q \in \mathcal{O}$. Since the robot follows $\mathcal{O}$ in a fixed direction, it must cross the line segment $[Q, T]$; at the crossing, T becomes a node of the LTG. Hence, the leaving condition will hold in this case. Consider next the case where T is not directly visible from any point of $\mathcal{O}$. Let C be a point on $\mathcal{O}$ which is closest to T. Let C' be the point on the neighboring obstacle which is the closest to C along the line segment $[C, T]$ (Fig. 14). During the boundary-following, the minimal distance to T satisfies $d_{min}(T) \geq d(C, T)$. Since T is reachable from P_i, it must be possible to move from C directly toward T. Hence $d(C', T) < d(C, T)$, and subsequently $d(C', T) < d_{min}(T)$. The leaving condition is then satisfied when C' becomes a node of the LTG. It therefore suffices to show that C' must become a node of the LTG.

For simplicity, assume that C' is not visible from the switch point P_i. The point C' is visible from the point $C \in \mathcal{O}$. Since the robot follows the boundary of $\mathcal{O}$ in a fixed direction, it must cross the line segment $[C, C']$. Hence C' must become visible during the boundary-following. The robot's path during this mode of motion can be partitioned as follows: either the robot moves in the interior of a straight-line segment which is the locally shortest path relative to $\mathcal{O}$, or it has just reached the endpoint of such a segment which is always a convex obstacle vertex. First, consider the case where the robot is moving along a straight-line segment. We may assume that C' lies in

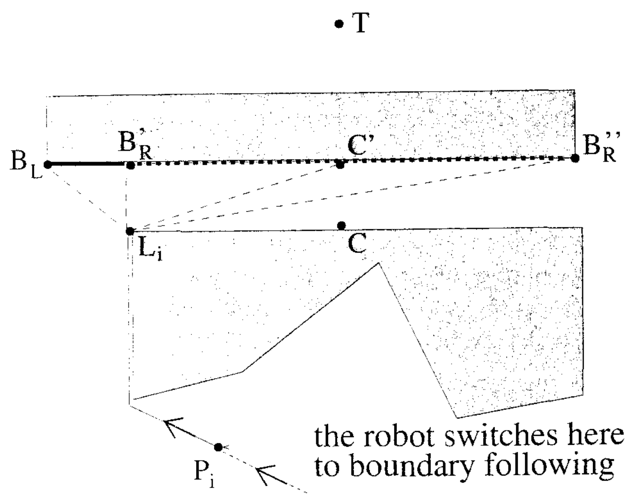

Fig. 14. When the robot reaches L_i, it simulates a smoothing of the corner while computing the LTG in all the intermediate directions. The node B_R scans the obstacle boundary from B'_R to B''_R, and C' becomes a node of the LTG during this scan.

the interior of a neighboring obstacle edge, which is denoted e. (If C' is an obstacle vertex, there exists a point C'' in the vicinity of C' which satisfies the leaving condition and is not a vertex.) The point C' can become visible in one of the following two ways. The first is when an occlusion node of the LTG slides along e until it reaches C'. At this point, C' becomes a node of the LTG. Also, C' can become visible when the robot crosses the extended edge associated with e, and this case is considered below.

Next, consider the case where C' becomes visible as the robot reaches a convex vertex of the obstacle, as illustrated in Figure 14. In this case, the visible set and hence, some of the LTG nodes, change discontinuously. However, according to the definition of TangentBug, the robot performs the following "smoothing" operation at the vertex. The robot assumes that the boundary's tangent vector changes continuously from the direction of the edge entering the vertex to the direction of the edge emerging from the vertex. The robot then computes the LTG for all the intermediate directions, and as a result the LTG nodes scan the boundary of the obstacles visible from the vertex. Since C' is visible from the vertex, it must become an LTG node during this scan.

Finally, consider the case where C' becomes visible when the robot crosses the extended edge associated with e. In this case, it is not necessarily true that C' becomes a node of the LTG when it first becomes visible. To see that C' does become a node of the LTG at some point, consider the event where C' *ceases* to be visible. (Since C' was initially invisible, it must return to being invisible before the robot completes a loop around the obstacle.) The crossing of the extended edge associated with e happens only once during the boundary-following. Hence, at the point where C' becomes invisible, either C' slides out of sight as an occlusion node of the LTG, or it disappears because the robot has just left a convex obstacle vertex. In both cases it must be that C' is a node of the LTG at the moment it disappears from sight. Thus, in all three cases C' becomes a node of the LTG, and the leaving condition becomes valid before the robot completes a loop around the obstacle. The leaving condition therefore becomes valid after a finite-length path. □

References

Arkin, R. C. 1987 (Raleigh, NC). Motor schema based navigation for a mobile robot: An approach for programming by behavior. *Proc. Int. Conf. Robotics and Automat.* Los Alamitos, CA: IEEE, pp. 264–271.

Bauer, R., Feiten, W., and Lawitzky, G. 1994. Steer angle field: An approach to robust maneuvering in cluttered, unknown environments. *Robot. Autonomous Sys.* 12:209–212.

Borenstein, J., and Koren, Y. 1990 (Cincinnati, OH). Real-time obstacle avoidance for fast mobile robots in cluttered environments. *Proc. Int. Conf. Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 572–577.

Choset, H., and Burdick, J. W. 1995 (Nagoya, Japan). Sensor-based planning, part II: Incremental construction of the generalized Voronoi graph. *Proc. Int. Conf. Robot. and Automat.* Washington, DC: IEEE, pp. 1643–1649.

Crowley, J. L., and Demazeau, Y. 1993. Principles and techniques for sensor data fusion. *Signal Processing* 32:5–27.

Foux, G., Heymann, M., and Bruckstein, A. 1993. Two-dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Trans. Robot. Automat.* 9(1): 96–102.

Goodridge, S. G., and Luo, R. C. 1994 (San Diego, CA). Fuzzy behavior fusion for reactive control of an autonomous mobile robot: Marge. *Proc. Int. Conf. Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 1622–1627.

Kamon, I., Rimon, E., and Rivlin, E. 1995. A new range-sensor based globally convergent navigation algorithm for mobile robots. Technical Report CIS-9517, Technion, Department of Computer Science, Haifa, Israel.

Khatib, O. 1985. Real-time obstacle avoidance for manipulators and mobile robots. *Proc. of the Int. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 500–505.

Leonard, J. J., and Durrant-Whyte, H. F. 1992. *Directed Sonar Sensing for Mobile Robot Navigation.* Boston: Kluwer Academic.

Liu, H., and Arimoto, S. 1992. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *Int. J. Robot. Res.* 11(4):376–382.

Lumelsky, V. J. 1991. A comparative study on the path length performance of maze-searching and robot motion planning algorithms. *IEEE Trans. Robot. Automat.* 7(1):57–66.

Lumelsky, V. J., and Skewis, T. 1990. Incorporating range sensing in the robot navigation function. *IEEE Trans. Sys. Man Cybernet.* 20(5):1058–1068.

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

Noborio, H. and Yoshioka, T. 1993. An on-line and deadlock-free path-planning algorithm based on world topology. *Proc. Conf. Intell. Robots and Sys., IROS.* IEEE/RSJ, pp. 1425–1430.

Reignier, P. 1994 (Grenoble, France). Molusc: An incremental approach of fuzzy learning. *Proc. of the Int. Symp. on Robot. Sys., IRS*, pp. 178–186.

Rencken, W. D. 1993. Concurrent localization and map building for mobile robots using ultrasonic sensors. *Proc. of the Conf. on Intell. Robot. and Sys., IROS.* IEEE/RSJ, pp. 2192–2197.

Rimon, E. 1997. Construction of C-space roadmaps from local sensory data. What should the sensors look for? *Algorithmica* 17(4):357–379.

Sankaranarayanan, A., and Vidyasagar, M. 1991 (Sacramento, CA). 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. *Proc. of the Int. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 1734–1941.

Stentz, A. 1994 (San Diego, CA). Optimal and efficient path planning for partially known environments. *Proc. of the Int. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 3310–3317.