I-Bug: An Intensity-Based Bug Algorithm

Kamilah Taylor and Steven M. LaValle

***Abstract*— This paper introduces a sensor-based planning algorithm that uses less sensing information than any others within the family of bug algorithms. The robot is unable to access precise information regarding position coordinates, angular coordinates, time, or odometry, but is nevertheless able to navigate itself to a goal among unknown piecewise-analytic obstacles in the plane. The only sensor providing real values is an intensity sensor, which measures the signal strength emanating from the goal. The signal intensity function may or may not be symmetric; the main requirement is that the level sets are concentric images of simple closed curves, i.e. topological circles. Convergence analysis and distance bounds are established for the presented approach.**

I. INTRODUCTION

Navigation in an unknown environment is a classical robotics problem. Typically, the robot must gather information about the obstacles in the environment, its position coordinates, orientation, and much more. If limited sensors deny the robot access to this information, one may wonder if it can complete any task of significance. Various portions of the radio wave spectrum are sensed by numerous devices, including submarines, wireless heart monitors, radios, televisions, mobile phones, and anything with bluetooth. The main question in this paper is: Can we get a robot navigate to the source of a transmitter among unknown obstacles while only being able to sense the signal intensity and estimate its local gradient? Yes we can.

To imagine the difficulty, suppose a woman is walking around a city trying to get to a tower. She can navigate around various buildings without knowing if she is in Kansas or Japan and without knowing her longitude and latitude. To make it more interesting, suppose she is blindfolded. There is not only uncertainty about position, but also about the surrounding environment. What information could she use to get to the tower? Suppose that the tower sends a signal, which could be a loud sound or a radio broadcast. Does there exist a strategy that enables her to successfully navigate to the tower based on signal intensity? What kinds of sensing and actuation are needed? This paper is motivated by such basic questions. Figure 1 shows an example of a simulated robot that executes a strategy based on incrementally maximizing a single intensity while moving among unknown obstacles and without knowing coordinates, orientation, or the signal mapping.

Our mathematical model falls mostly into the well-known family of bug algorithms, which have two main modes of movement: following obstacle boundaries and moving towards the goal. The original bug algorithms [14], [15] proposed a minimalist sensing model and a robot navigation algorithm to bring the robot to a specified goal in a 2D environment with unknown smooth obstacles. The work was later extended to include a range sensor, which led to improved bounds on the total distance traveled [13]. Since then there have been several other bug algorithms. In [7], the TangentBug was proposed, which enhanced the sensing model to improve the bound on the length of the path to the goal. In [8], TangentBug was extended to three dimensions. WedgeBug and its relative RoverBug [9], [10], [11] restrict the TangentBug sensing model so that it can be applied in an actual planetary rover. A bug algorithm for solving pursuit-evasion was presented in [16].

Fig. 1. The robot starts at the lower-left green dot and moves towards the upper-right red dot while traversing various obstacle boundaries. Level sets of equal intensity are represented by the circular arcs.

The motivation for our paper came from carefully studying the models of previous bug algorithms. Even though these models are aimed at minimizing sensing and mapping requirements, they appear to require some precise information that may not be necessary. For example, the original Bug1 and Bug2 algorithms [15] use: 1) a contact sensor, 2) coordinates of the initial robot position, 3) coordinates of the current robot position, 4) coordinates of the target, and 5) odometry to obtain the distance traveled around an obstacle boundary. This information is evident when studying the particular approach. Bug1 goes around the entire obstacle, calculates the closest leaving point, returns to that point, and then goes in a straight line towards the target. Bug2 calculates an "m-line", which is a line segment that connects the initial point to the goal point, and always moves on that line unless it is contacted an obstacle. While moving along an obstacle, it follows the boundary until it is once more on the m-line, and then it returns to moving towards the target on the m-line. Thus, it seems that the robot needs a position sensor, a linear

K. Taylor is with the Department of Computer Science, University of Illinois at Urbana-Champaign ktaylo21@illinois.edu

S. Lavalle is with the Department of Computer Science, University of Illinois at Urbana-Champaign lavalle@uiuc.edu

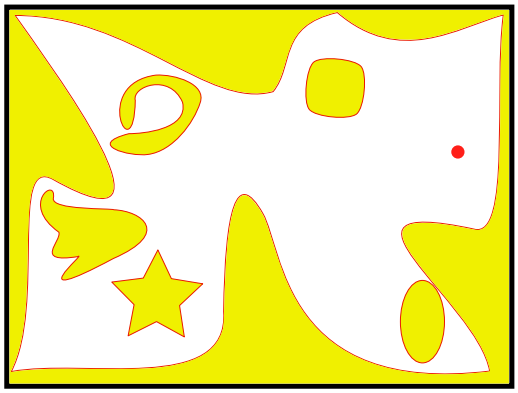

Fig. 2. There may be an outer obstacle O_{outer}, which has a finite-length boundary curve but extends infinitely outward in all directions.

odometer, and angular odometer to execute both the Bug1 and Bug2 algorithms. Bug2 would also need to calculate whether the obstacle has intersected the m-line. VisBug's algorithm[15] is based on the Bug2 algorithm but uses a range sensor to decrease the Bug2 path bound. TangentBug uses a 360° range sensor to avoid following the boundary and instead move a certain distance away from the boundary unless it is unavoidable. WedgeBug is based on TangentBug and uses a more limited 30° to 45° range sensor to minimize the number of sensor readings.

A common theme among the bug algorithms is that they rely on knowing the robot's exact coordinates. Our motivation comes from a simple observation: the previous bug robots had access to the exact coordinates of every place they visited, and in some cases to the exact distance they traveled. If they had had unlimited memory, it would even have been possible to reconstruct a perfect map of their environments. It seems that it ought to be possible to navigate through an environment without collecting all of this information. We therefore want to determine whether the robot can navigate to a goal without collecting all of this information. Can it reach the goal without having any coordinates?

Section II introduces a coordinate-free mathematical model for a bug that navigates based on the intensity of a signal emanating from the goal. Section III presents a solution for the case of a radially symmetric intensity function. The resulting plan guarantees that the robot reaches the goal and an upper bound on the total distance traveled is given. Section IV addresses the more general case of an asymmetric intensity function.

II. PROBLEM FORMULATION

Suppose that a point robot moves in $\mathbb{R}^2$ according to a kinematic differential drive model. A differential drive robot consists of two independently controlled wheels attached to an axle. By sending equal power to both identical wheel motors, the robot can easily *move straight* or *rotate in place*. Therefore, we use these two motions as basic primitives to control the robot.

Let $\mathcal{O}$ be a set of *obstacles*, in which each $O \in \mathcal{O}$ is closed with a connected piecewise-analytic boundary that is finite in length. Furthermore, the obstacles in $\mathcal{O}$ are pairwise-disjoint. There may be a countably infinite number of obstacles; however, at most a finite number are contained in any fixed disc (this property is called locally finite in [15]). The obstacle set $\mathcal{O}$ may contain an *outer obstacle* O_{outer} that is unbounded; all other obstacles are bounded. See Figure 2.

Let E be the closure of $\mathbb{R}^2$ minus all $O \in \mathcal{O}$ and be called the *environment*. Note that the environment is connected and may or may not be bounded.

A point called the *tower* exists at some location $(x_t, y_t) \in \mathbb{R}^2$. The tower broadcasts a *signal*, which is modeled as an intensity function over $\mathbb{R}^2$. Let m denote the *signal mapping* $m : \mathbb{R}^2 \to [0, 1]$, in which $m(p)$ yields the *intensity* at $p \in E$, generated from the tower. It is assumed that the maximum intensity, 1, is achieved at the tower: $m(x_t, y_t) = 1$. If the robot is at p, then the intensity is translated accordingly as $m(p - (x_t, y_t))$. Due to translational symmetry and the fact that the robot never receives position coordinates, it may be assumed without loss of generality that $(x_t, y_t) = (0, 0)$. This significantly simplifies calculations, and we can rephrase the problem as: The robot must find the origin $(0, 0)$, which contains the tower.

For any $i \in [0, 1]$, consider the level sets (or preimages)

$$m^{-1}(i) = \{p \in \mathbb{R}^2 \mid m(p) = i\}. \tag{1}$$

We want to allow intensity functions that are as complicated as those measured in practice from radio signals or other physical sources. An important restriction, however, will be that we allow only one local maximum, which is at the tower. In spite of this, it will be assumed that m could be any locally Lipschitz, piecewise-analytic function for which $m^{-1}(i)$ is homeomorphic to a circle, i.e. a topological circle, for every $i \in (0, 1)$ and $m^{-1}(1) = \{(0, 0)\}$ (m may define, for example, some polyhedral surface). Furthermore, the level sets must be concentric, with $(0, 0)$ at the center. We make a general position assumption that for the boundary of every $O \in \mathcal{O}$ and every preimage $m^{-1}(i)$, they are either disjoint or intersect in a finite number of places. Let M denote the set of all intensity functions that satisfy these conditions.

Let $M_s \subset M$ denote the set of all *radially symmetric* intensity functions. In this case, the level sets form concentric circles in the classical sense (rather than concentric topological circles). As an example,

$$m(p) = \frac{1}{p_x^2 + p_y^2} \tag{2}$$

causes the intensity to decay quadratically with distance, without regard to direction. This is a common idealized model for radio transmission. More generally, if the level sets are not concentric circles, then $m \in M \setminus M_s$ is called *asymmetric*.

The environment E and even the signal mapping m are unknown to the robot. Furthermore, the robot does not even know its own position and orientation. Based on these quantities, a *state space* X is defined as

$$X \subset SE(2) \times \mathcal{E} \times M \tag{3}$$

in which $SE(2)$ is the set of all possible robot positions and orientations, $\mathcal{E}$ is the set of all possible environments, and M is the set of all possible intensity mappings.

Each sensor available to the robot will be defined as a mapping $h : X \to Y$ from the state space X into an *observation space* Y. Three main sensors will be considered. First, the *contact sensor* indicates whether the robot is touching the environment boundary ∂E:

$$h_t(x) = \begin{cases} 1 & \text{if } x \in \partial E \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The other two sensors obtain information regarding the tower. The *intensity sensor* indicates the strength of the signal from position p:

$$h_i(x) = h(p, \theta, E, m) = m(p). \quad (5)$$

The robot can use the intensity sensor to determine when it is at the tower, which uniquely occurs when $h_i(x) = 1$. However, if the robot does not know the maximum possible intensity, then a "tower detection sensor" can be added; this is avoided for this paper since the two become mathematically equivalent.

For the third sensor, there are two possibilities. The *tower alignment sensor* indicates whether the robot is facing the tower:

$$h_a(x) = \begin{cases} 1 & \text{if } \theta = \text{atan2}(-p) \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Alternatively, the *gradient alignment sensor* indicates whether the robot is facing the direction of steepest ascent of m:

$$h_a(x) = \begin{cases} 1 & \text{if } (\cos\theta, \sin\theta) \propto \nabla m(p) \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

At nonsmooth points, the gradient ∇ is assumed to be extended in a standard way from nonsmooth analysis; see [4] in general, and [3] for the use of this in the context of sensor-based planning. In this general case, $h_a(x) = 1$ if $(\cos\theta, \sin\theta)$ is proportional to any vector in the *generalized gradient* [4]:

$$\text{co}\left\{\lim_{i\to\infty} \nabla m(p_i) : p_i \to p,\ m'(p_i) \text{ exists}\right\}, \quad (8)$$

in which the p_i correspond to any sequence that converges to p, co denotes the convex hull, amd m' denotes the derivate of m. Intuitively, this definition gathers up all possible gradients by taking derivatives along all sequences convering to p for which derivatives exist.

In Section III, m is radially symmetric, in which case either alignment sensor can be used because they give the same result. In Section IV, the asymmetric case is handled, and only the gradient alignment sensor is used. The robot has no other sensors, such as global positioning, odometry, or a compass. Therefore, it is unable to obtain precise position or angular coordinates.

Now consider possible actions or *motion primitives* that are given to move the robot. Each motion primitive must terminate on its own using sensor information. The robot is allowed only three motion primitives:

u_{fwd} The robot goes straight forward in the direction it is facing, stopping only if: 1) it contacts the obstacle ($h_t(x) = 1$), 2) hits the tower ($h_i(x) = 1$), 3) detects a local maximum in intensity along its line of motion.

u_{ori} The robot rotates counterclockwise, stopping only when it is aligned with the tower ($h_a(x) = 1$).

u_{fol} The robot travels around an obstacle boundary counterclockwise, maintaining contact to its left at all times, stopping only when it reaches a local maximum in the intensity.

There are obviously some hidden details regarding how these primitives are implemented, especially in the cases of u_{fwd} and u_{fol}. Both of these have termination conditions that depend on detecting a local maximum. This could be achieved in practice by sampling the intensity at high frequency and checking the relations $i_{k-1} > i_{k-2}$ and $i_{k-1} \geq i_k$, of the last three intensity observations are i_{k-2}, i_{k-1}, and i_k. Relaxing the comparison between i_{k-1} and i_k to include the possibility of equality allows the robot to detect plateaus in intensity values. This sampling policy could obviously cause the robot to slightly pass the maximum, which could be deemed to be insignificant due to a high sampling rate, or the robot could execute a short reversal motion. Furthermore, for u_{fol}, the robot must be able to move itself along the wall using the contact sensor. This might be achieved by mounting a horizontal wheel that rolls along the wall and is force controlled. Such details will not be considered further; however, it is important to point out that some subtle details remain regarding the implementation of the primitives in practice. In this paper, the primitives are given, and seem reasonable under the sensing model.

PLAN FOR THE SYMMETRIC CASE

1) Let $i_L = h_i(x)$.
2) Apply u_{ori} and then u_{fwd}.
3) If $h_i(x) = 1$, then terminate; the tower was reached.
4) If $i_L \neq h_i(x)$, then let $i_H = h_i(x)$.
5) Apply u_{fol}.
6) If $h_i(x) > i_H$ then go to Step 1.
7) Go to Step 5.

Fig. 3. A successful plan for the case of a radially symmetric intensity function.

III. RADIALLY SYMMETRIC CASE

The section presents a plan for the robot that guarantees it will reach the tower after a finite number of primitives have been applied. In this section, u_{fwd} always terminates when either the tower or boundary is reached; the possibility of a local maximum in intensity arises only in Section IV.

A. A plan for the I-Bug

Using its motion primitives and enough memory to store two intensity values, i_L and i_H, the plan is shown in Figure 3. The intensity i_H is the intensity observed when the current obstacle was contacted via completion of a u_{fwd} motion. The intensity i_L is the value obtained just prior to the execution of u_{fwd}. This is used in Step 4 to compare with the current intensity $h_i(x)$ to determine whether u_{fwd} caused the robot to move. If the robot moved, then a new value for i_H is

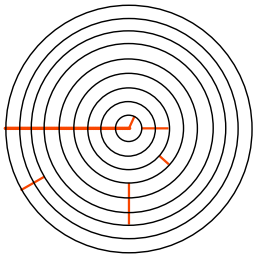

Fig. 4. When path segments are linearly rotated, their sum is equivalent to a single path.

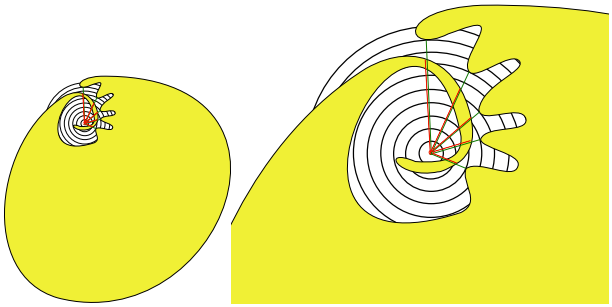

Fig. 5. The robot is repeatedly sent around the obstacle before finally reaching the goal.

stored because the robot moved across the interior of E. It is assumed that the starting position lies in the interior of E, which guarantees that i_H is defined in the first iteration. In each execution of Step 5, the robot moves to another local maximum, and then it tries to leave the boundary in Step 6 if the maximum is greater than i_H. If after u_{ori} the robot is facing the boundary, then it cannot make progress, and $i_L = i_H$. This indicates that another local maximum must be reached before trying to escape again. Note that our robot cannot follow the Bug1 approach in [15] because the robot is unable to determine whether it has traveled completely around the obstacle.

B. Convergence

Does this plan actually succeed? The following lemma represents a crucial step in establishing convergence to the tower:

Lemma 1: For every obstacle boundary ∂O and every possible tower location $\in \mathbb{R}^2 - O$, there exists at least one intensity local maximum $p \in \partial O$ for which the disc centered at the tower $(0, 0)$ with radius $\|p\|$ is disjoint from the interior of O.

Most proofs have been omitted to save space. The complete version is available on-line.

Convergence is established in the following proposition:

Proposition 2 (Convergence): The plan in Figure 3 causes the robot to reach the tower after a finite number of steps, regardless of the particular environment E, initial robot position in the interior of E, and tower location in E.

Proof: After executing Step 2 for the first time, either the tower is reached or the robot contacts the boundary of an obstacle. Assuming the latter, Step 4 stores i_H, the intensity at this boundary point. The main idea of the proof is that the intensity increases monotonically with every subsequent execution of Step 2. Since distance decreases monotonically as intensity increases, the robot arrives at $(0, 0)$. Step 6 ensures that u_{fwd} is attempted only at a point that is closer to $(0, 0)$ than the point at which the robot arrived at the obstacle boundary (where it recorded i_H). It might seem that an infinite loop is possible by failure to satisfy the condition of Step 6 or by the motion being blocked by the obstacle boundary. However, Lemma 1 ensures that it is always possible to leave the obstacle boundary and obtain a higher intensity value. In the worst case, the robot may repeatedly return to the same obstacle boundary ∂O, but it cannot become trapped. Each time it arrives at O, i_H is larger, and the number of local maxima is finite. A new departure point along ∂O exists each time due to Lemma 1. Eventually, the robot must leave from a global intensity maximum, and its direction faces the interior of the disc D from the proof of Lemma 1. Therefore, the robot is not blocked, it increases the intensity, and will never contact ∂O again. Since this is assured for every obstacle, the robot must eventually arrive at $(0, 0)$. ∎

Note that in the proof above, the robot does not necessarily know whether it is returning to the same obstacle multiple times. It may alternate between several obstacles unknowingly, but this causes no harm.

C. Bounding the total distance

How far might the robot travel in the worst case to reach the tower? Let $\ell(p_0, E)$ denote the distance traveled by the robot after executing the plan in Figure 3 from position p_0. This would be the reading obtained by a perfect odometer, if it had existed. Let N be the total number of obstacles that intersect a disc of radius $\|p_0\|$, centered at $(0, 0)$. A local maximum at a point $p \in \partial O$ is called *unblocked* if the robot can freely move toward the tower from p, without immediately entering the interior of O. The following proposition bounds the total distance traveled:

Proposition 3 (Bounding the Path Length): The total distance traveled by the robot satisfies the bound:

$$\ell(p_0, E) = \ell_{fwd} + \ell_{fol} \leq \|p_0\| + \sum_{k=1}^{N} n_k c_k, \qquad (9)$$

in which n_k is the number of unblocked local maxima along O_k, c_k is its perimeter, ℓ_{fwd} is due to u_{fwd}, and ℓ_{fol} is due to u_{fol}.

Note that if all obstacles are convex, then the second term of (9) can be improved to $\ell_{fol} \leq \sum_k^N P_k$. Paths arbitrarily close to this worst-case behavior exist, as shown in Figure 6.

It is interesting that the bound in Proposition 3 is similar to that of Bug2 [15], even though our robot receives much less information. In that case, the bound on ℓ_{fol} is $1/2$ of what is obtained in (9).

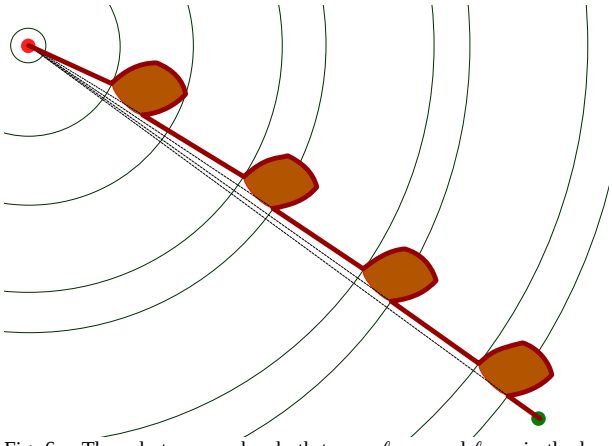

Fig. 6. The robot approaches both terms, ℓ_{fwd} and ℓ_{fol}, in the bound in the case of convex obstacles.

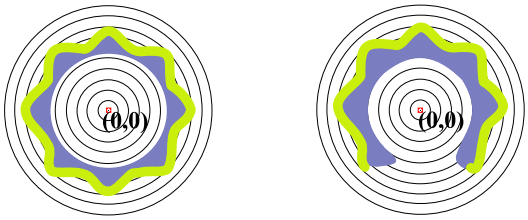

Fig. 7. The environments above are two representatives from the sequence of obstacles that cause k local maxima. The highlighted region of the obstacles indicates regions where the robot collects identical sensor readings. The robot does not know how many peaks may exist. Furthermore, it cannot determine whether there is a solution beyond the next peak, as would occur for the example on the bottom. Since the environment is unknown, the robot cannot decide whether the tower is reachable.

D. Decidability

It has been assumed so far that the tower lies in E. Suppose that the tower may lie anywhere in $\mathbb{R}^2$ and the robot must either move to the tower if it exists or declare after a finite number of steps that the tower is unreachable. Not only does the plan in Figure 3 fail to achieve this, the following proposition establishes that the robot cannot generally decide whether $(0,0) \in E$:

Proposition 4 (Decidability): Using its sensors and motion primitives, it is impossible for the robot to determine whether the tower is reachable, in other words whether $(0,0) \in E$.

The main impediment with the robot deciding when the tower is reachable is that it cannot tell when it returns to the same point along ∂O. This is a familiar problem in the searching of unknown mazes [2], graphs [1], [5], and polygons [6]. The usual solution is to introduce a *pebble* that serves as a marker. There are many ways to simulate the effect of a pebble, but all of them require additional sensor or actuation capabilities.

E. Obstacles Without Nonsmooth Points

Suppose that we restrict the obstacle boundaries to be analytic, rather than piecewise-analytic. This implies that every point along ∂O has a well-defined normal, in the sense from classical calculus. Now remove the tower alignment sensor and convert u_{fwd} into a new primitive u_{nor} that always moves toward the tower in the direction of the normal at the robot position in ∂O. Suppose that the plan in 3 is modified by executing u_{nor} in Step 2, instead of u_{ori} followed by u_{fwd}.

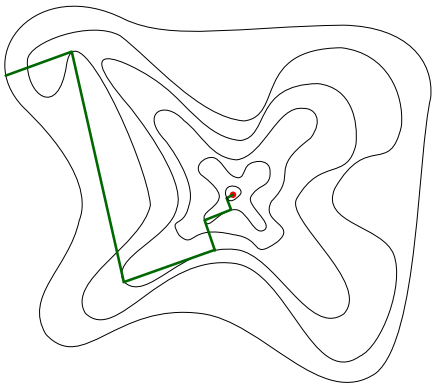

Fig. 8. Robot makes progress towards the target in an environment with asymmetric intensity.

PLAN FOR THE ASYMMETRIC CASE

1) Let $i_L = h_t(x)$.
2) Apply u_{ori} and then u_{fwd}.
3) If $h_i(x) = 1$, then terminate; the tower was reached.
4) If $i_L \neq h_i(x)$, then let $i_H = h_i(x)$.
5) If $h_t(x) = 0$, then go to Step 1.
6) Apply u_{fol}.
7) If $h_i(x) > i_H$ then go to Step 1.
8) Go to Step 6.

Fig. 9. A successful plan for the case of an asymmetric intensity function. The difference is that multiple iterations are needed when crossing the interior of E. This is reflected in Step 5, which did not exist in Figure 3.

Proposition 5: If the boundary of every obstacle is analytic, then the modified plan (which avoids the tower alignment sensor) always succeeds and the path satisfies the bound in Proposition 3.

IV. GENERAL ASYMMETRIC CASE

This section generalizes some of the ideas from Section III to the setting of intensity functions that are asymmetric. In this section, the level sets are topologically equivalent to circles, but may take any shape. The intensity function m is piecewise-analytic with a single maximum at $(0,0)$. The primary trouble caused by this case is that the gradient of the intensity function no longer "points" to the tower. In the symmetric case, the tower alignment sensor (6) and gradient alignment sensor (7) produce the same orientation. In this section, the two sensors generally produce different results. It is assumed here that the gradient alignment sensor is used. Note that a straight-line motion will no longer take the robot to the tower. Fortunately, the robot is able to make progress by relying on the main idea from the classical optimization technique *steepest descent with line searching* (SDLS) [12].

The plan from Figure 3 is modified in the present setting to obtain the plan shown in Figure 9. The only real difference is given by the insertion of Step 5. During the execution of u_{fwd}, the robot may fail to reach the obstacle boundary. Therefore, it must realign itself and move in a new direction. Figure 8 shows a sample path in the asymmetric intensity scenario. This iteration continues until the tower or boundary

is reached. If the boundary is reached, then u_{fol} is applied as in Section III.

The proof of convergence follows the same general strategy as in Section IV. Recall Lemma 1, which was perfect for ensuring that the robot does not get trapped moving along an obstacle boundary. In the current setting, replace the disc $D((0,0),p)$ with a *topological disc*, $B((0,0),p)$, which is defined as

$$B((0,0),p) = \{p' \in E \mid m(p') \geq m(p)\}. \qquad (10)$$

Informally, the topological disc includes all points with intensity greater than or equal to the intensity at p. Using this definition, the following lemma can be stated, which generalizes Lemma 1 to a topological disc:

Lemma 6: For every obstacle boundary ∂O and every possible tower location, there exists at least one intensity local maximum $p \in \partial O$ for which the topological disc $B((0,0),p)$ is disjoint from the interior of O.

Using Lemma 6, it is straightforward to establish the convergence of the plan in Figure 9:

Proposition 7 (Convergence): For any $\epsilon > 0$, the plan in Figure 9 causes the robot to arrive within ϵ distance from the tower after a finite number of steps, regardless of the particular environment E, initial robot position in the interior of E, and tower location in E.

It is natural to wonder whether a bound can be constructed on the total path length, as established in Proposition 3. In the current setting, the second term of $\sum_{k=1}^{N} n_k c_k$ remains as an upper bound on the motions due to u_{fol}. The first term, however, is complicated by the convergence rate of the SDLC iterations, which depends on the properties of the intensity function m. For optimization problems, conjugate gradient descent is usually preferred over SDLC because of its faster convergence rate; however, our robot does not receive enough information to apply the method.

V. CONCLUSIONS

We have successfully determined that a robot can, in theory, navigate to a signal source using its intensity and gradient, in spite of having no other information about its configuration, the obstacles, or even the intensity mapping (except that its level sets are topological circles). The robot uses a contact sensor, an intensity sensor, and an alignment sensor to achieve the task of reaching a goal, which is the signal source. The robot does not have access to perfect clocks, odometry, or other sensors that would enable it to infer any coordinates in $\mathbb{R}^2$, its orientation in $[0, 2\pi)$, or its total distance traveled. For a radially symmetric intensity function, we presented a plan for I-Bug that is guaranteed to succeed in a finite number of steps. Also, a bound is provided on the total distance traveled. If the intensity function is asymmetric, the plan is slightly modified, but nevertheless converges. This case seems quite interesting due to its extreme generality. Although the robot is unable to move directly to the goal, its convergence is assured through an approach that is mathematically equivalent to the steepest descent line search algorithm from optimization.

Many interesting questions remain for future research. It is surprising that the robot can accomplish the task without being aware of whether it is returning to the same obstacles. What other tasks can be accomplished in spite of this confusion? What tasks require distinguishability between obstacles? What forms of sensing should be added to give the robot enough information to make such distinctions (e.g., a mathematical pebble)? In another direction, can the plans given in this paper be improved by allowing the robot to alternative between clockwise and counterclockwise directions? Could a randomized approach lead to good expected-case behavior? Another interesting direction is to navigate using intensity-based coordinates with a robot among multiple towers.

Acknowledgments

This work was supported by the DARPA SToMP program (DSO HR0011-07-1-002). The views expressed in this paper are not necessarily endorsed by DARPA. The authors thank Stephen Bond for a helpful discussion.

REFERENCES

[1] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings Annual Symposium on Foundations of Computer Science*, 1998.

[2] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proceedings Annual Symposium on Foundations of Computer Science*, pages 132–142, 1978.

[3] H. Choset and J. Burdick. Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):126–148, 2000.

[4] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Springer-Verlag, Berlin, 1998.

[5] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, December 2005.

[6] B. Gfeller, M. Mihalak, S. Suri, E. Vicari, and P. Widmayer. Counting targets with mobile sensors in an unknown environment. In *ALGOSENSORS*, July 2007.

[7] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics & Automation*, 13(6):814–822, December 1997.

[8] I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[9] S. L. Laubach and J. W. Burdick. An autonomous sensor-based path-planning for planetary microrovers. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[10] S. L. Laubach and J. W. Burdick. Practical autonomous path planner for turn-of-the-century planetary microrovers. *Proceedings of SPIE*, 3525:182, 1999.

[11] S. L. Laubach and J. W. Burdick. RoverBug: Long Range Navigation for Mars Rovers. *LECTURE NOTES IN CONTROL AND INFORMATION SCIENCES*, pages 339–348, 1999.

[12] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Wiley, New York, 1973.

[13] V. J. Lumelsky and T. Skewis. A paradigm for incorporating vision in the robot navigation function. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 734–739, 1988.

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

[15] Vladimir Lumelsky. *Sensing, Intelligence, Motion: How Robots and Humans Move in an Unstructured World*. Wiley-Interscience, 2005.

[16] S. Rajko and S. M. LaValle. A pursuit-evasion bug algorithm. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1954–1960, 2001.