2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Egress: An online path planning algorithm for boundary exploration

K.R. Guruprasad and Prithviraj Dasgupta

***Abstract*—We consider the problem of navigating a mobile robot that is located at any arbitrary point within a bounded environment, to a point on the environment's outer boundary and then, using the robot to explore the perimeter of the boundary. The environment can have obstacles in it and the location and size of these obstacles are not provided *a priori* to the robot. We present an online path planning algorithm to solve this problem that requires very simple behaviors and computation on the robot. We analytically prove that by using our algorithm, the robot is guaranteed to reach and explore the outer boundary of the environment within a finite time.**

I. Introduction

Online path planning for mobile robots is an essential technique for navigating the robots within their environment. Previous research in this area approaches the online path planning problem as finding an obstacle-free trajectory from the current location of the robot to a goal point or target. Several approaches such as bug algorithms [1], [2], potential functions [3], [4], have been proposed to address this problem. In all these approaches, the target is specified as a stationary point and the robot's goal is to find a path to reach the target. In this paper, we address a complementary problem that we call the *egress problem* - how can a robot that is placed within an environment with obstacles, navigate to a set of connected points such as the perimeter of its environment's boundary. The number and location of obstacles in the environment is not known *a priori* to the robot and it is possible that some of these obstacles might partially occupy or occlude the environment's boundary. In such situations, a pre-determined target point on the boundary might be unreachable by the robot. Therefore, it makes sense to study the egress problem so that the robot can find an 'escape route' to reach the boundary of its environment.

The solution to the egress problem can be useful for several robotic domains. For example, in robotic perimeter patrolling applications [5], [6], a robot that is deployed at any arbitrary point within the environment can use it to find a path to the perimeter of its environment. In several, robotic area or terrain coverage algorithms [7], [8], [9], [10], the region to be covered is partitioned so that each robot can cover its allocated sub-region. However, a robot only knows the geometric boundary of its sub-region and is not aware of the physical boundary of the sub-region. The robot can use the solution to the egress problem to explore this boundary before covering the inside of the sub-region. To the best of our knowledge, any strategy for the exploration of the boundary of a region is not available in the literature.

We approach the egress problem in two parts. In the first part called the *ReachBoundaryBasic* algorithm, the robot moves towards the environment's boundary along radial lines while wall-following around obstacles it encounters. However, a robot using this strategy might be unable to reach the boundary for certain cases of complex exit routes. We then provide an escape route algorithm called *EscapeLoop* that explores only specific portions of the robot's path traced while it was performing ReachBoundaryBasic, to find a path to the environment's outer boundary. We prove the completeness of the proposed algorithm, provide an upper bound on the trajectory length and show that the algorithms are guaranteed to terminate in finite time.

II. Problem Formulation

Consider a convex and compact set $Q \in \mathbb{R}^2$. The boundary ∂Q of Q, referred to as the *geometric boundary*, is a Jordan (that is, a closed simple) curve in $\mathbb{R}^2$. Define $I_N = \{1, 2, \ldots, N\}$. Let $O = \{O_1, O_2, \ldots O_K\}$, be a set of a finite number of obstacles with each obstacle $O_i \in \mathbb{R}^2$ being a connected, but not necessarily convex, closed set. Also, $O_i \cap Q \neq \emptyset, \forall i \in I_K$, that is, no obstacle is completely outside Q. Let $O_{in} = \{O_i | O_i \cap \partial Q = \emptyset, i \in I_K\} \subset O$ be the set of *internal obstacles*, and $O_{bound} = O \setminus O_{in}$ be the set of *non-internal obstacles*, or, the obstacles which extend beyond Q. The free space $Q_{free} = (Q \setminus O)$ is assumed to be connected and the boundary $\partial(Q_{free})$ of Q_{free} is called the *outer boundary*.

We consider a wheeled robot that is represented as a disc of diameter D_r. Its task is to explore the outer boundary. Let $X(t)$ and $\phi(t)$ represent the position and orientation of robot at time t with $X(0) \in Q_{free}$. The configuration of the robot at time t is denoted by $(X(t), \phi(t))$. The robot is equipped with limited-range proximity sensors for detecting obstacles, and is able to follow a wall on its left or right side using these sensors. To avoid collisions with a wall or an obstacle, the robot has to maintain a clearance of d_{cl}. To maintain this clearance the robot is represented as a virtual disc of diameter $D = D_r + 2d_{cl}$. The robot also has a localizing device such

This work has been supported as part of the COMRADES project supported by the U.S. DoD Office of Naval Research, grant no. N0000140911174.

K.R Guruprasad is an Assistant Professor at the Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore, 57025, India, and Visiting Postdoctoral Researcher at the Department of Computer Science, University of Nebraska, Omaha, NE 68182, USA. krgprao@gmail.com

Prithviraj Dasgupta is an Associate Professor at the Department of Computer Science, University of Nebraska, Omaha, NE 68182, USA. pdasgupta@mail.unomaha.edu

The authors would like to thank Corey Spitzer for Webots implementation of the algorithm's code.

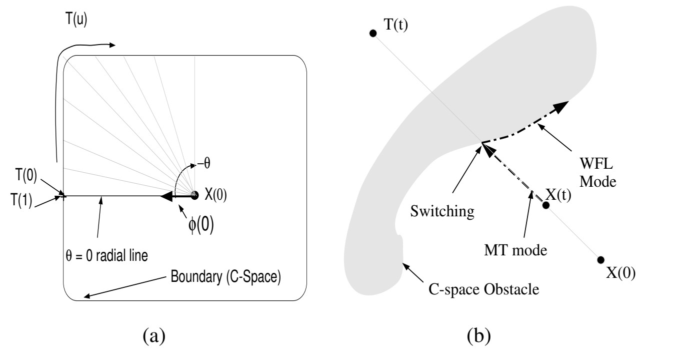

Fig. 1. (a) $\partial(^C Q)$ is parameterized based on $X(0)$ and $\phi(0)$. The point $T(u(\theta))$ corresponding to angle θ is the point of intersection of the radial line and $\partial(^C Q)$, with $u = -\theta/(2\pi), \theta \in [0, -2\pi]$. As $\partial(^C Q)$ is a closed curve, $T(0) = T(1)$. (b) Illustration of switching from MT mode to WFL mode. The robot path is shown in '-.-' line.

as a GPS or an IR-based positioning system that allows it to localize itself within a global frame of reference.

Egress Problem: Given a bounded environment Q with initially unknown obstacles O within it, a robot with a start location $X(0)$ and heading $\phi(0)$ within Q, find a path for the robot to reach the outer boundary of Q and explore it.

To simplify the analysis of the egress problem, we consider the robot inside its configuration space (C-space) [11]. In a path planning problem, only the position of a robot needs to be specified and its orientation is implicitly assumed to be tangential to the path. Therefore, the virtual disc of diameter D corresponding to the robot in the workspace is represented as a point in the C-space. To represent a space $W \subset \mathbb{R}^2$ from the workspace in the C-space, we use the notation $^C W$. Obstacles expand in the C-space; consequently, $O_i \subset {}^C O_i$, and, as a result, $^C(Q_{free}) \subset (Q_{free})$.

Assumption 1: For every point $x \in \partial(^C(Q_{free}))$, there exists a circle of diameter $\delta : \delta > 0$, which is tangential to $\partial(^C(Q_{free}))$ at x and not intersecting $\partial(^C(Q_{free}))$.

This ensures that no two portions of $\partial(^C(Q_{free}))$ are incident on each other.

Parametrization: Consider a line from the robot's initial position $X(0)$ along its heading direction $\phi(0)$, intersecting $\partial(^C Q)$ at point $T(0)$. We call this line a radial line and let $\theta = 0$ for this radial line as shown in Figure 1(a). Now $\partial(^C Q)$ can be parameterized by rotating this radial line anchored at $X(0)$ clockwise from $\theta = 0$ to $\theta = -2\pi$ (or counterclockwise with $\theta \in [0, 2\pi]$). Consider $T(u) \in \partial(^C Q), u \in [0, 1]$ as the parametrization of $\partial(^C Q)$ with $u = -\theta/(2\pi)$ as the parameter and with $X(0)$ and $\phi(0)$ as reference. The assumption that Q is convex and bounded ensures that any radial line from any point in Q intersects ∂Q, and hence, $\partial(^C Q)$, only once. This makes $T : [0, 1) \to \partial(^C Q)$ a bijection. As the robot moves, θ changes with t, and hence u. With a slight abuse of notation we use $\theta(t)$ and $u(t)$, to refer to value of θ and u at time t. Further, we use $T(t)$ to refer to $T(u(t))$.

Mode	Description
Move toward target (MT)	Move toward a target point T along a straight line at a constant speed $v > 0$
Wall follow left (WFL) or Wall follow right (WFR)	Follow a wall/boundary, on left or right side, at a constant speed $v > 0$

TABLE I
BASIC BEHAVIORS OF THE ROBOT

III. The path planning algorithm

We have divided the solution of our problem into two parts: i) find a path to reach any point on the outer boundary, and ii) use a simple wall/boundary following algorithm to explore the outer boundary. In certain cases, obstacles might occupy the entire $\partial(^C Q)$. In such a situation, the robot will not be able to judge if it has reached a point on $\partial^C(Q_{free})$.

The table I shows the basic behaviors of the robot used in the path planning algorithm.

A. ReachBoundaryBasic (RBB) algorithm

We describe a path planning algorithm called *ReachBoundaryBasic* (RBB) which attempts to make the robot reach a point on $\partial(^C Q)$, if $\partial(^C Q) \cap (^C Q_{free}) \neq \emptyset$, without guaranteeing it. The basic path planning algorithm has several steps, and we discuss them below:

1) Initialization: Based on $X(0)$ and $\phi(0)$, $\partial(^C Q)$ is parameterized and $T(0) \in \partial(^C Q)$ is computed, as illustrated in Figure 1(a).

2) MT Mode: The robot moves from $X(t)$ toward $T(t)$, along a radial line starting at $X(0)$. Thus, there is no change in $\theta(t)$ (refer to the Figure 1(a)), and hence $u(t)$ and $T(t)$ are fixed in this mode. The path followed by the robot in MT mode is called a radially outward path (ROP).

3) Switching from MT to WFL (MT2WFL): While the robot is in MT mode, if it encounters an obstacle before reaching $T(t)$, it switches to WFL mode. Geometrically, this happens when the radial line from $X(0)$ to $T(t)$ intersects an obstacle boundary in C-space, as illustrated in Figure 1(b).

4) WFL mode: The robot moves while following the wall of the obstacle on its left side. The path followed by the robot in this mode is referred to as a tangential path (TP). While moving in this mode, the robot computes $\theta(t)$ which is the angle between the radial lines joining $X(0)$ to $T(0)$ and $X(0)$ to $X(t)$. If at some time interval $[t_1, t_2]$, $\dot{\theta}(t) < 0$, and in time interval $[t_2, t_3]$, $\theta(t)$ initially increases and then decreases, and $\theta(t_2) = \theta(t_3)$, then u remains fixed in the time interval $[t_2, t_3]$. This ensures that $\dot{u}(t) > 0, \forall t > 0$, that is, the target point $T(u)$ only moves in clockwise direction along $\partial(^C Q)$. This is illustrated in Figure 2(a). In discrete time, the parameter update is done as follows:

$$u(t) = \begin{cases} -\theta(t)/2\pi, \text{if} -\theta(t)/2\pi > u(t-1) \\ u(t-1), \text{otherwise} \end{cases} \quad (1)$$

After updating u the target point $T(u(t))$ is updated.

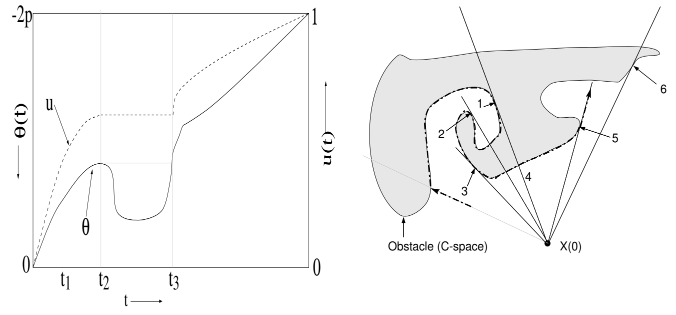

Fig. 2. (a) The parameter u is updated as θ changes. However, in the time interval $(t_2, t, 3]$, $u(t)$ remains constant to ensure that $\dot{u} \geq 0$. (b) Switching from WFL mode to MT mode can occur at a point along the obstacle boundary (in C-space), where a (radial) line from $X(0)$ is tangential to it. At points marked $1, 2, 3, 5$ and 6, this condition is satisfied. However the condition of heading direction being same as the direction of radially outward line $(T(t) - X(0))$ is satisfied only at points marked 5 and 6. This is due to the fact that u remains constant as robot moves (in WFL mode) from 1 to 4. At point 6, which is point of inflection, robot switches from WFL mode to MT mode, and immediately to WFL mode.

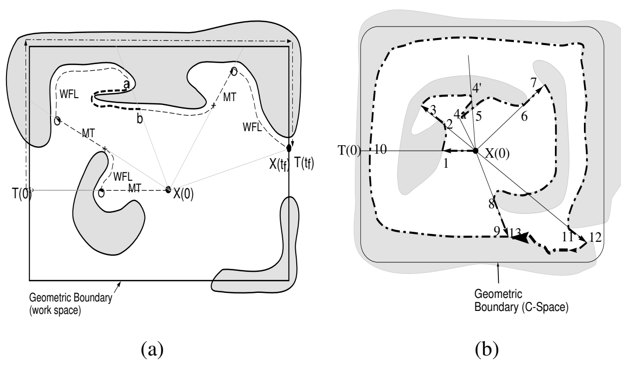

Fig. 3. The path is shown by dashed line. The transition points from MT to WFL are shown by 'O's and those from WFL to MT are shown by '+'s along the path. The path of $T(t)$ is shown in '-.-'. (a) Robot successfully reaches the outer boundary (shown in work space). Target does not move while the robot is moving along a part of the path shown in thick dashed line (a-b), as $u(t)$ is not updated as it should be a non-decreasing function. (b) The robot gets into a loop with the RBB algorithm (shown in C-space).

5) Switching from WFL to MT (WFL2MT): While in WFL mode, if the robot's heading direction is aligned with the vector $(T(t) - X(0))$, the robot switches to MT mode and moves toward $T(t)$. This switching can happen only when u is increasing (or $u(t) > u(t-1)$). The process is illustrated in Figure 2(b). In addition (not shown in figure), if a part of boundary (in C-space) is tangential to the radial line $(T(t) - X(0))$ (or equivalently $(X(t) - X(0))$), then the robot switches from WFL mode to MT mode. In this particular case, the paths with WFL and MT modes correspond to each other.

6) Termination: The RBB algorithm terminates and makes the robot stop when

i) The robot reaches a point on $\partial({}^C Q)$. This corresponds to successful situation.
ii) $X(t_2) = X(t_1)$ for some $t_2 > t_1 > 0$, that is, the robot encounters its own previous path, making it a cyclic path or a loop. This corresponds to failure to reach a point on the outer boundary.

Figure 3(a) shows an illustrative path with the RBB algorithm. The points marked '+' along the robot's path are the points where the robot switched from WFL to MT mode, and points marked 'O' are points where the robot switched from MT to WFL mode. A portion of the path is shown darker (a-b), where the value of θ initially decreases and then increases such that its value is same at the end points of this part of the path. Thus, there is no change in $u(t)$ when the robot is moving in this path. This ensures that the robot continues in the WFL mode and does not does not switch to MT mode. The robot successfully reached a point on the outer boundary in this case. Figure 3(b) shows another scenario where the robot gets into a loop.

B. Analysis of RBB algorithm

The RBB algorithm will be analyzed here.

Definition 1: A right (left) extreme point 2_i (1_i), relative to $X(0)$, on $\partial({}^C O_i), O_i \in O_{in}$ is the first point at which a radial line from $X(0)$ touches $\partial({}^C O_i), O_i \in O_{in}$, and any other radial line from $X(0)$ which touches (intersects) $\partial({}^C O_i), O_i \in O_{in}$ makes a positive (negative) acute angle with line joining $X(0)$ and 2_i (1_i).

Figure 4 shows the extreme points. Note that: There is exactly one right extreme point and exactly one left extreme point for each $O_i \in O_{in}$; The radial line corresponding to an extreme point is tangential to $\partial({}^C O_i)$ if $\partial({}^C O_i)$ is at least C^1 continuous curve (that is, first derivative is continuous) at the corresponding extreme point. If at an extreme point $\partial({}^C O_i)$ has discontinuity in its first derivative, then there are two derivatives one from left and one from right. If the radial line corresponding to an extreme point touches more than one point (or line segments) on $\partial({}^C O_i)$, then the first point (starting from $X(0)$) is the extreme point.

Lemma 1: The robot following obstacle $O_i \in O_{in}$ always switches from WFL mode to MT mode and leaves O_i at the right extreme point on $\partial({}^C O_i)$.

Proof is fairly straightforward and is not provided here. [1]

Lemma 2: The robot path generated by the RBB algorithm is a simple curve (that is, it does not intersect itself).

Proof. If the path intersects itself, one of the following should happen: i) Two radially outward paths (ROP) intersect each other; ii) Two tangential paths (TP) intersect each other; iii) A ROP and a TP intersect each other.

It is intuitive that no two ROPs can intersect each other, as each of them is a segment of radial line starting from $X(0)$.

The TPs are parts of path around obstacles (in C-space) or two segments of same TP. By Assumption 1, it is clear that no two TPs will intersect or be tangentially incident upon each other. Only if the robot started in the WFL mode, it can

[1] In a discrete time implementation of RBB algorithm, it is possible that the robot crosses the right extreme point before checking for switching condition, and hence misses switching from WFL mode to MT mode. Special care must be taken to avoid such a situation.

reach $X(0)$ at some time $t_1 > 0$, say, while moving in same the direction. Once $X(t_1) = X(0)$, it is easy to see that for $t > t_1$, the robot path will retrace the cyclic path it traced during time interval $[0, t_1]$, forming a simple closed curve.

Geometrically, a ROP and a TP can intersect each other. Let $X(t_1) = X(t_2)$ for some $t_2 > t_1 \geq 0$. That is, at t_2, the robot encounters its path at a point which it had reached at t_1. At $X(t_1)$ the robot could have been either in the WFL or the MT mode, or might have switched between the modes. As $X(t_2) = X(t_1)$ it is obvious that the RBB algorithm makes the robot take the same decision at times t_2 and t_1. That is, after $t > t_1$, the robot retraces the cyclic path it traced during time interval $[t_1, t_2]$. Thus, the path can not intersect itself, and hence it is a simple curve. $\square$

The implication of Lemma 2 is that, if the robot path generated by the RBB algorithm touches itself, and it is not stopped, then it will move in a cyclic path or a loop forever. However, at this instance RBB terminates by stopping the robot. It can be observed that the path gets into a loop if all the outward radial lines from $X(0)$, which are tangential to their corresponding TPs where the robot should switch from WFL to MT mode, intersect obstacles. The robot can escape from obstacles only in MT mode, and each of the ROPs intersects an obstacle, thus, it gets into a loop. This condition is purely geometric and depends on $X(0)$ and obstacle configuration.

Remark 1: The robot can touch its path only when it is in WFL mode, and if it did not start in WFL mode, it can touch only a ROP. Thus, the robot need not keep track of its entire path to identify a loop. It needs to only remember the points along the path where switching between modes had occurred and $X(0)$.

Lemma 3: A robot starting in Q_{free} will not go out of Q_{free}.

Proof. Let us assume $X(0) \in {}^{C}Q_{free}$ and at some time $t_o > 0$, $X(t_o) \notin {}^{C}Q_{free}$. As the path is continuous (C^0), $\exists t_1 \in (0, t_o), s.t. X(t_1) \in \partial({}^{C}Q_{free})$. However, RBB terminates at t_1 stopping robot at $X(t_1) \in \partial({}^{C}Q_{free})$. Hence, $X(t) = X(t_1), \forall t \geq t_1$, contradicting our assumption that $X(t_o) \notin {}^{C}Q_{free}$. $\square$

Lemma 4: The algorithm RBB does not lead to a path that encircles an obstacle or a set of obstacles.

Proof. Let $\partial({}^{C}O_i)(a, b)$ represent the part of $\partial({}^{C}O_i)$ starting at a ending at b. The extreme points divide $\partial({}^{C}O_i)$ into two parts. With $\partial({}^{C}O_i)(1_i, 2_i)$ being the closer part of the boundary and $\partial({}^{C}O_i)(2_i, 1_i)$ being the farther part of the boundary to $X(0)$.

The robot can only encounter an obstacle in MT mode from a point which is in between line joining $X(0)$ and any point in $\partial({}^{C}O_i)(1_i, 2_i)$, reaching a point on the closer boundary where it switches to WFL mode. If the robot is located on or beyond the farther boundary, then robot will move away from the obstacle, and will never encounter it. Thus, no TP can start from the farther part of boundary. Without loss of generality, we may assume that the robot encounters obstacle O_i and switches to WFL mode at $1' \in \partial^{C}O_i(1_i, 2_i)$.

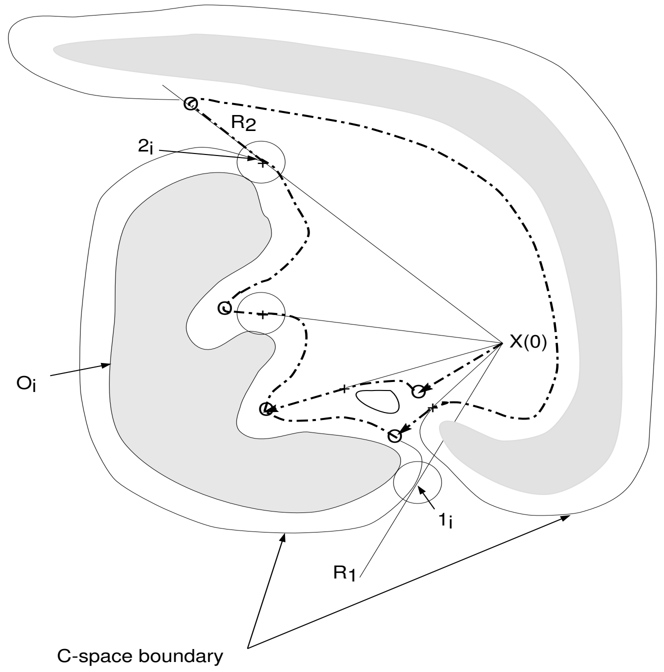

Fig. 4. Illustration of bound on path length. Points marked 1_i and 2_i are the left and right extreme points of O_i respectively.

Now, by Lemma 1, the robot will switch from WFL mode to MT mode and leave the obstacle at the right extreme point 2_i. Thus, there cannot be any path along the farther part of boundary. Hence, robot cannot encircle an obstacle or a set of obstacles. $\square$

Theorem 1: The RBB algorithm generates a path of bounded length.

Proof. As discussed in Lemma 4, the maximum length of TP following the boundary of an obstacle O_i is the length of $\partial({}^{C}O_i)(1_i, 2_i)$. Let $L(\cdot)$ denote the length of a curve, and $L(TP_i)$ be the length of the path when the robot encounters obstacle O_i. This path includes path while robot is in WFL mode moving along $\partial({}^{C}O_i)$ and ROPs starting and ending on $\partial({}^{C}O_i)$ (see Figure 4). Therefore,

$$L(TP_i) < L(\partial({}^{C}O_i)(1_i, 2_i)) < L(\partial({}^{C}O_i)) \qquad (2)$$

Thus, the total path length, $L(TP_T)$ of all TPs (before the robot reaches boundary or the path gets into a loop) is

$$L(TP_T) < \sum_{i=1}^{K} L(\partial O_i) \qquad (3)$$

The result is true for any starting location $X(0)$. It is obvious that for any ROP $(ROP) \leq \text{Dia}(Q)$. Now, we look at the number of ROPs. As illustrated in Figure 4, it is easy to see that the robot can enter (or encounter) $\partial({}^{C}O_i)$ through a ROP only once after it leaves $\partial({}^{C}O_i)$. Further, once the robot enters $\partial({}^{C}O_i)$, it can leave only once. A leaving ROP is either an entering ROP for another obstacle or the last ROP leading to a point on $\partial({}^{C}Q)$. Thus, there can be a maximum of two entering ROPs per obstacle. Hence, maximum number of ROPs is $2K+1$ (One ROP leading to $\partial({}^C Q)$), and the total length $L(ROP_T)$ of all ROPs is

$$L(ROP_T) \leq (2K+1)\text{Dia}(Q) \tag{4}$$

Now as the combined length of both ROPs and TPs is bounded, the total path length is bounded. □

Corollary 1: The RBB algorithm makes the robot stop in finite time.

Proof. The path length by the RBB algorithm is bounded by Theorem 1 and the robot speed is constant. Thus, time taken by the robot to travel a finite path length is finite. □

Lemma 5: The robot path using the RBB algorithm is cyclic if and only if it does not reach a point on $\partial({}^C Q)$.

Proof. Let us assume that the robot path using the RBB algorithm is cyclic. This happens when the robot encounters its own path, that is, $X(t_2) = X(t_1)$ for some $t_2 > t_1 > 0$. This is possible only if the robot has not reached any point on $\partial({}^C Q)$ at any time $t' \in [0, t_2]$. Otherwise, the robot would have stopped at t'. The robot cannot reach any point on $\partial({}^C Q)$ at any time $t > t_2$ as the RBB algorithm makes the robot stop at $t = t_2$. Even if the robot continues using the RBB algorithm, it traces the cyclic path it traced in the time interval $[t_1, t_2)$.

Now consider the case when the robot does not reach a point on $\partial({}^C Q)$. Let us assume that the robot does not encounter its own path, and hence does not get into a cyclic path. The robot stops only when it has either reached a target point on $\partial({}^C Q)$, or has encountered its own path thus, making the path a closed curve. This implies that the robot never stops, which contradicts Corollary 1. Thus, if the robot does not reach a point on $\partial({}^C Q)$, then it will re-encounter its own path leading to a cyclic path and then stop. □

The implication of Lemmas 4 and 5 is that if the robot gets into a loop, and at least one point on $\partial({}^C Q)$ is reachable, then there is some path by which the robot can come out of the loop. In the following section, we describe an algorithm for the robot to identify an 'escape route' out of such a loop.

C. EscapeLoop (EL) Algorithm

Here we provide an algorithm called EscapeLoop which attempts to move the robot out of the cyclic path generated by the RBB algorithm, when it failed to reach the outer boundary. Once the robot gets into a cyclic path, it moves along the same path until it finds the first point, where a switching from MT to WFL mode had occurred. At this point, robot switches to the WFR mode (in contrast to switching to WFL in RBB) and looks for an escape from the loop, radially outward. While moving in WFR mode, $u(t)$, and hence $T(t)$, is updated only when $\theta(t)$ increases. While in WFR mode, if the robot's heading direction is along the vector $T(t) - X(0)$, it switches to MT mode. If the robot fails to escape the loop, it will continue along the path generated by the RBB algorithm along the ROP it started from. The robot continues moving along this path until it reaches the next point, where a switching from MT to WFL modes had occurred. This process continues until either the robot escapes from the loop by switching from WFR mode to MT mode, or it exhausts all points where a switching from MT to WFL modes using the RBB algorithm had occurred.

Lemma 6: The algorithm EL is unsuccessful if and only if $\partial({}^C Q) \cap^C (Q_{free}) = \emptyset$, that is, no point on the geometric boundary belongs to the free space, and the current cyclic path determined by the RBB algorithm has TPs coinciding with a portion of $\partial^C(Q_{free})$.

Proof. Let $\partial({}^C Q) \cap^C (Q_{free}) = \emptyset$, and the current cyclic path determined by the RBB algorithm has portions of TPs coinciding with a portion of $\partial^C(Q_{free})$. By Lemma 3 the robot can not come out of Q_{free}.

Now consider a situation when robot failed to escape from the loop using the algorithm EL. There are two possibilities: first, the cyclic path is made up of only TPs, and second, there is at least one ROP in the cyclic path. In the first situation it is obvious that the cyclic path itself is the outer boundary. In the second case, if the robot can escape from the loop, it is only at points where it has switched from WFL mode to MT mode. If at all these switching points, the robot reaches back to the loop by following WFR, then it is clear that the TPs of the loop coincide with the outer boundary. □

D. ReachBoundary Algorithm

The ReachBoundary algorithm combines the RBB algorithm and the EL algorithm to make the robot reach a point on the outer boundary. The robot starts with the RBB algorithm. If it reaches a point on the outer boundary, the RBB algorithm terminates successfully. If the robot gets into a loop with the RBB algorithm, it tries to escape from the loop using the EL algorithm. If the robot is successful in escaping the loop, it continues with the RBB algorithm. The robot continues to switch between RBB and EL algorithms until either it reaches a point on the outer boundary, or, it is unsuccessful to escape the loop with the EL algorithm. If the EL algorithm is unsuccessful, then any point on a TP of the loop is a point on the outer boundary. Thus, the robot successfully reaches a point on the outer boundary. This completeness result is stated below:

Theorem 2: The ReachBoundary algorithm ensures that the robot reaches a point on $\partial({}^C Q_{free})$ in finite time.

Proof. The proof follows from Lemmas 5 and 6.

E. ExploreOuterBoundary Algorithm

With the ReachBoundary algorithm, the robot reaches a point in $\partial({}^C Q_{free})$. Now the robot can follow the outer boundary in WFL (or equivalently WFR) mode, without switching to MT mode to complete the exploration. A flowchart for the ExploreOuterBoundary algorithm is given in Figure 5.

Theorem 3: A robot using the ExploreOuterBoundary algorithm successfully explores the outer boundary of the free space in finite time.

Proof. Theorem 2 guarantees, that with the ReachBoundary algorithm, the robot successfully reaches a point on $\partial({}^C Q_{free})$ in finite time. Now by moving either in WFL or in WFR mode, the robot successfully explores the outer boundary in finite time. □

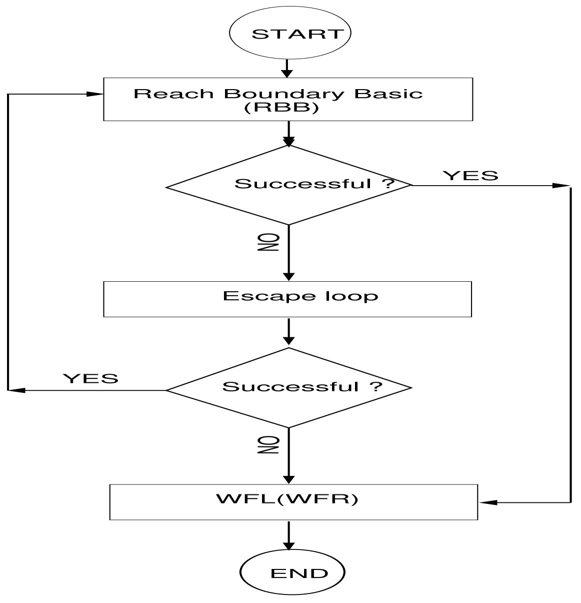

Fig. 5. Explore Outer Boundary algorithm

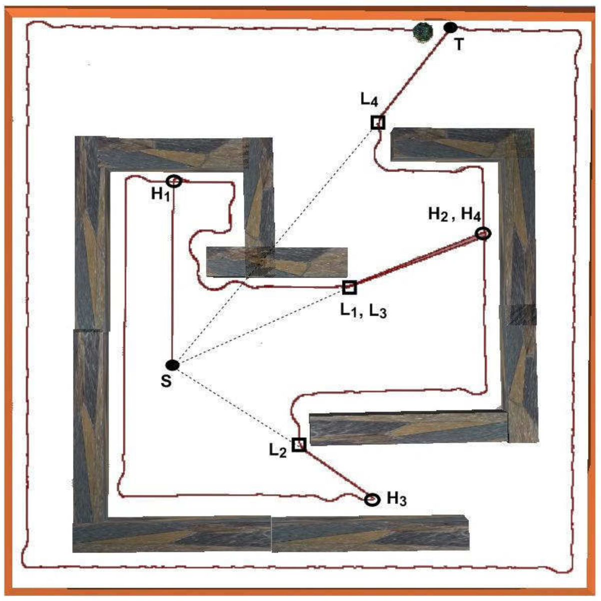

Fig. 6. A snapshot from Webots illustrating the ExploreOuterBoundary algorithm. using the EL algorithm. The hit points are shown with circles and leaving points by squares.

IV. Illustrative simulation result

We implemented the ExploreOuterBoundary algorithm on the Webots robot simulator using the model of an e-puck robot. For localizing the robot, we used a GPS node available within Webots[2]. Figure 6 shows a snapshot of a scenario where the robot starts at S. The robot path with the RBB algorithm is $S \rightarrow H_1 \rightarrow L_1 \rightarrow H_2 \rightarrow L_2 \rightarrow H_3 \rightarrow H_1$. The robot escapes this loop successfully using the EL algorithm at H_4 (which is the same as H_2) by moving in WFR mode and leaves the obstacle at L_4 to finally reach a point T on the outer boundary. Once the robot reaches T, it moves in the WFL mode to explore the outer boundary.

V. Conclusions

We described a novel problem of exploration of the outer boundary of a geometrically known space occupied by obstacles that are not known *a priori* by a robot. A path planning algorithm called RBB was presented which either makes the robot reach a point on the outer boundary successfully, or gets it into a cyclic path if unsuccessful. An algorithm called EL was used to make the robot escape from the cyclic path. By using RBB and EL algorithms, a ReachBoundary algorithm was presented which guarantees that the robot reaches a point on the outer boundary of the free space. Finally an ExploreOuterBoundary algorithm was presented exploring the outer boundary. It was proved analytically that by using the proposed algorithm, the robot is guaranteed to reach and explore the outer boundary of the environment within a finite time. The proposed algorithm was implemented on the Webots robot simulator using the model of an e-puck robot. As part of future work, we plan to address issues related to robustness to localization errors, and implement the algorithm on real e-puck robots.

[2]For implementation with physical e-puck robots, either an overhead camera or odometry data can be used.

References

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

[2] I. Kamon, E. Rimon, and E. Rivlin, TangetBug: a range-sensor-based navigation algorithm, *International Journal of Robotics Research*, vol. 19, no. 9, 1998, pp. 934–953.

[3] T. Zhang, Y. Zhu, and F. Song, Real-time motion planning for mobile robots by means of artificial potential field methods in unknown environment, *Industrial robot: An International Journal*, vol. 37, no. 4, 2010 pp. 384-400.

[4] S.S. Ge and Y.J. Cui, New potential functions for mobile robot path planning, *IEEE Trans. on Robotics and Automation*, vol. 16, no. 5, 2000, pp. 615–620.

[5] N. Agmon, S. Kraus, and G.A. Kaminka, Multi-robot perimeter patrol in adversarial settings, *Proc of IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 2008, 2339-2345.

[6] N. Basilico, N. Gatti, and F. Amigon, Leader-follower strategies for robotic patrolling in environments with arbitrary topologies, *Proc of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, 2009, 57-64.

[7] H. Choset, Coverage for robotics - A survey of recent results, *Annals of Mathematics and Artificial Intelligence*, vol. 31, 2001, pp. 113–126.

[8] M. Schneider-Fontan and M. Mataric, Territorial multi-robot task division, *IEEE Trans on Robotics and Automation*, vol. 15, no. 5, 1998, pp. 815–822.

[9] K.R. Guruprasad, Z. Wilson, and P. Dasgupta, Complete coverage of an initially unknown environment by multiple robots using Voronoi partition, *Proc of 2nd International Conference on Advances in Control and Optimization in Dynamical Systems*, Bengaluru, India, February 16-18, 2012.

[10] I. Rekleitis, A.P. New, E.S. Rankin, and H. Choset, Efficient boustrophedon multi-robot coverage: an algorithmic approach, *Annals of Mathematics and Artificial Intelligence*, vol. 52, 2008, pp. 109–142.

[11] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion-Theory, Algorithms, and Implementation*, The MIT Press, Cambridge, Massachusetts, 2005.