An exploration strategy by constructing Voronoi Diagrams with provable completeness

Jonghoek Kim, Fumin Zhang, and Magnus Egerstedt

Abstract— We present novel exploration algorithms and a control law that enable the construction of Voronoi diagrams over unknown areas using a single autonomous vehicle equipped with range sensors. Our control law and exploration algorithms are provably complete. The control law uses range measurements to enable tracking Voronoi edges between two obstacles. Exploration algorithms make decisions at vertices of the Voronoi diagram to expand the explored area until a complete Voronoi diagram is constructed in finite time. MATLAB simulation results are provided to demonstrate the effectiveness of both the control law and the exploration algorithms.

I. INTRODUCTION

This paper addresses the problem of exploring an unknown workspace using one autonomous vehicle equipped with range sensors. We assume that the vehicle has the ability to determine a point on obstacle boundary that is closest to the vehicle. We call such a point the *closest point*. Two closest points may appear on two sides of the vehicle. The path that has equal distances from these closest points is the Voronoi edge. All Voronoi edges form the Voronoi diagram that reveals the topological structure of the workspace. If the vehicle visits all Voronoi edges in the workspace, then we consider the workspace as being completely explored.

Voronoi diagrams have been used in areas such as computational geometry [1]–[3], VLSI design [4], and sensor networks [5]–[7]. In robotics, Voronoi diagrams have been utilized to obtain paths that satisfy minimum clearance requirements [8]–[11]. Voronoi diagrams can be generalized into higher dimensions, and also fit a wide class of robots [12], [13]. Several extensions of Voronoi diagrams have been developed by other researchers, including the generalized Voronoi graph (GVG), the Hiearchical GVG (HGVG) and the Reduced GVG (RGVG) in [12], [14]–[16][1]. Exploration of an unknown workspace by incrementally constructing the Voronoi diagram was achieved in [8], [13], [17]. But completeness of the algorithms has NOT been proved.

We develop a Voronoi edge tracking control law that is provably convergent. This law is based on the shape dynamics derived in [18]. In [18] and [19], a gyroscopic feedback control law was developed to control the interaction

Jonghoek Kim, Fumin Zhang, and Magnus Egerstedt are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, USA. Email: jkim37@mail.gatech.edu, {fumin,magnus}@ece.gatech.edu

[1]The HGVG connects the disconnected GVG in three dimensional enviroment [15]. The RGVG is the resultant GVG after removing Voronoi edges connected to the obstacle boundaries [16]. We construct Voronoi diagrams similar to the RGVG in that no Voronoi edge is connected to the obstacle boundaries.

between the vehicle and a closest point so that the vehicle follows the obstacle boundary either to its left or its right. This controller design method was generalized to cooperative motion patterns on closed curves for multiple vehicles in [20], [21], and extended to the design of pursuit-evasion laws in three dimensions [22]. The closest point was also used for path following in [23]. Our curve tracking control law extends previous work by using information from the closest points on both sides of the vehicle. This results in a tracking behavior of the Voronoi edge between two obstacles.

Utilizing the Voronoi edge tracking behavior, we develop provably complete exploration algorithms, denoted as *Boundary Expansion (BE) algorithms*, which enable the construction of a topological map based on Voronoi diagrams[2]. Although many results exist in literature regarding the construction of Voronoi diagrams, to our knowledge, BE algorithms are unique with provable completeness over a compact workspace.

BE algorithms are composed of two algorithms, denoted by Algorithm 1 and Algorithm 2. Applying Algorithm 1, the trajectory of a vehicle constructs a simple closed curve that encloses an obstacle to its right. Then, using Algorithm 2, the vehicle iteratively expands the explored area in a way that one obstacle is added to the area at a time. In this way, the vehicle constructs a Voronoi diagram by "expanding" the explored area in discrete and finite steps.

Using the BE algorithms, the vehicle is controlled so that it does not trace back along the Voronoi edge that the vehicle has just traversed. In other words, the vehicle keeps moving forward and does not stop or back track at any intersection. Hence the vehicle is able to operate at higher speed. This feature makes the BE algorithms suitable for fast moving vehicles, such as unmanned aerial vehicles.

The paper is organized as follows. Section II introduces necessary background regarding Voronoi diagrams and the workspace of interest. Section III introduces a provably convergent control law to track Voronoi edges. Section IV discusses BE algorithms. Section V provides proofs for the convergence of BE algorithms. Section VI demonstrates simulation results, and Section VII provides conclusions.

II. THE WORKSPACE AND ITS VORONOI DIAGRAM

Consider a connected and compact workspace $W \subset \mathcal{R}^2$ whose boundary, ∂W, is a regular curve. Let $O_1, O_2, ... O_{M-1}$ be $M-1$ disjoint, and compact obstacles such that $O_i \subset W$.

[2]Due to space limitations, our algorithms in this paper are not presented in standard format, but rather described in English to increase readability.

O_M is a "virtual" obstacle that bounds the workspace, i.e., $\partial W \subset \partial O_M$. We denote the set of obstacles S_O by $S_O = \{O_1, O_2, ..., O_M\}$.

We define the *Voronoi cell* for an obstacle O_i as the set of points that are closer to O_i than to any other obstacle in S_o for $i = 1, 2, ..., M$ i.e.

$$V(O_i) = \{q \in W | \min_{z \in O_i} \|z - q\| < \min_{z' \in O_i'} \|z' - q\|,$$
$$\forall O_i' \in S_O \setminus O_i\}, \tag{1}$$

where $\| \cdot \|$ is the Euclidean norm in $\mathcal{R}^2$. $\partial V(O_i)$ is the boundary of the Voronoi cell for O_i, i.e., $V(O_i)$. Also, $\overline{V}(O_i) = V(O_i) \bigcup \partial V(O_i)$. The *Voronoi diagram* of the workspace is defined as the union of all cell boundaries [3] i.e.

$$D(W) = \bigcup_{O_i \in S_O} \partial V(O_i). \tag{2}$$

The shared boundary of two Voronoi cells is a *Voronoi edge*. More specifically, a Voronoi edge between two Voronoi cells $V(O_i)$ and $V(O_j)$ is defined by

$$E_{ij} = \partial V(O_i) \bigcap \partial V(O_j). \tag{3}$$

III. TRACKING ONE VORONOI EDGE

In this section, we develop a feedback control law to make a vehicle, with its dynamics approximated by a unit speed particle, move along a Voronoi edge. The feedback control law uses measurements at two closest points on the right and the left hand side of the vehicle as seen in Fig.1.
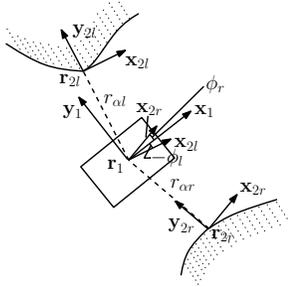

Fig. 1. A vehicle with boundary curves to the left and to the right of the vehicle.

A. Shape Dynamics

In Fig. 1, $\mathbf{r}_1$ denotes the position of the vehicle, and $\mathbf{x}_1$ denotes the heading direction of the vehicle. $\mathbf{r}_{2r}$ is the closest point to the right of the vehicle, and $\mathbf{x}_{2r}$ denotes the unit tangent vector to the boundary curve at $\mathbf{r}_{2r}$. Also, ϕ_r is the angle measured clockwise from $\mathbf{x}_{2r}$, which is the tangent vector at $\mathbf{r}_{2r}$, to $\mathbf{x}_1$, the heading direction of the vehicle. The relative position between the vehicle and the closest point to the right of the vehicle is $\mathbf{r}_{\alpha r} = \mathbf{r}_{2r} - \mathbf{r}_1$, and $r_{\alpha r} = \|\mathbf{r}_{\alpha r}\|$.

Similarly, $\mathbf{r}_{2l}$ is the closest point to the left of the vehicle, and $\mathbf{x}_{2l}$ denotes the unit tangent vector to the boundary curve at $\mathbf{r}_{2l}$. Also, ϕ_l is the angle measured clockwise from $\mathbf{x}_{2l}$, which is the tangent vector at $\mathbf{r}_{2l}$, to $\mathbf{x}_1$, the heading direction of the vehicle. The relative position between the vehicle and

the closest point to the left of the vehicle is $\mathbf{r}_{\alpha l} = \mathbf{r}_{2l} - \mathbf{r}_1$, and $r_{\alpha l} = \|\mathbf{r}_{\alpha l}\|$.

We choose the positive directions of the boundary curves such that

$$\begin{aligned} \mathbf{x}_1 \cdot \mathbf{x}_{2l} &= \cos(\phi_l) > 0, \\ \mathbf{x}_1 \cdot \mathbf{x}_{2r} &= \cos(\phi_r) > 0, \end{aligned} \tag{4}$$

which means that $-\pi/2 < \phi_l < \pi/2$ and $-\pi/2 < \phi_r < \pi/2$.

Consider the boundary curve to the right of the vehicle, the shape dynamics are given by [18] as follows.

$$\dot{r}_{\alpha r} = -\sin(\phi_r), \tag{5}$$

$$\dot{\phi}_r = (\frac{\kappa_r}{1 - \kappa_r r_{\alpha r}}) \cos(\phi_r) - u, \tag{6}$$

where κ_r denotes the curvature of the boundary at the closest point to the right of the vehicle. Similarly, for the boundary curve to the left, we have

$$\dot{r}_{\alpha l} = \sin(\phi_l), \tag{7}$$

$$\dot{\phi}_l = (\frac{\kappa_l}{1 + \kappa_l r_{\alpha l}}) \cos(\phi_l) - u, \tag{8}$$

where κ_l denotes the curvature of the boundary at the closest point to the left of the vehicle.

B. Tracking Control and Convergence Analysis

In this section, we design a tracking control law based on Lyapunov function. Consider the Lyapunov function candidate

$$V = -\ln(\cos(\frac{\phi_l + \phi_r}{2})) + \lambda (r_{\alpha l} - r_{\alpha r})^2, \tag{9}$$

where $\lambda > 0$ is a constant. In (9), the term $-\ln(\cos(\frac{\phi_l + \phi_r}{2}))$ penalizes misalignment between the heading direction of the vehicle and the tangent vector to the Voronoi edge. The term $r_{\alpha l} - r_{\alpha r}$ in (9) makes the vehicle converge to the Voronoi edge. The time derivative of V is

$$\begin{aligned} \dot{V} &= \tan(\frac{\phi_l + \phi_r}{2})(\frac{1}{2}((\frac{\kappa_r}{1 - \kappa_r r_{\alpha r}}) \cos(\phi_r) \\ &+ (\frac{\kappa_l}{1 + \kappa_l r_{\alpha l}}) \cos(\phi_l) - 2u) \\ &+ 4\lambda (r_{\alpha l} - r_{\alpha r}) \cos(\frac{\phi_r + \phi_l}{2}) \cos(\frac{\phi_l - \phi_r}{2})), \tag{10} \end{aligned}$$

where we have used shape dynamics (5),(6),(7), and (8). Also, $\sin(\phi_l) + \sin(\phi_r) = 2 \sin(\frac{\phi_l + \phi_r}{2}) \cos(\frac{\phi_l - \phi_r}{2})$ is applied. We design steering control u so that $\dot{V} \le 0$.

One choice of u that leads to $\dot{V} \le 0$ is

$$\begin{aligned} u &= \frac{1}{2}((\frac{\kappa_r}{1 - \kappa_r r_{\alpha r}}) \cos(\phi_r) + (\frac{\kappa_l}{1 + \kappa_l r_{\alpha l}}) \cos(\phi_l)) \\ &+ 2\lambda (r_{\alpha l} - r_{\alpha r})(\cos(\phi_l) + \cos(\phi_r)) \\ &+ \mu \sin(\frac{\phi_l + \phi_r}{2}), \tag{11} \end{aligned}$$

where $\mu > 0$ is a constant. The time derivative of V in (10) with u given by (11) is

$$\dot{V} = -\mu \frac{\sin^2(\frac{\phi_l + \phi_r}{2})}{\cos(\frac{\phi_l + \phi_r}{2})} \le 0. \tag{12}$$

where (4) is used. Thus, $\dot{V} \leq 0$ and $\dot{V} = 0$ if and only if $\sin(\frac{\phi_l + \phi_r}{2}) = 0$. But by (4), we see that $\dot{V} = 0$ if and only if $\phi_l + \phi_r = 0$.

Theorem 1: Suppose that $1 + \kappa_l r_{\alpha l} \neq 0$ and that $1 - \kappa_r r_{\alpha r} \neq 0$. Then, using the steering control law in (11), the unit speed vehicle, whose initial position satisfies (4), converges to the state where it moves along a Voronoi edge[3].

IV. BE Algorithms

We construct the Voronoi diagram of W using one vehicle by controlling the vehicle so that it traverses all Voronoi edges E_{ij} for $i, j = 1, 2, ..., M$.

A. Definitions and Assumptions

We define an *intersection P* as a point at which the following conditions are satisfied:

1) there exists a circle centered at the point P intersecting obstacle boundaries at more than two points. These points are called the *closest points* at the intersection[4].
2) the interior of the circle does not intersect any obstacles. The circle is called an *intersection circle*, and illustrated in Fig.2.

The lines connecting the intersection and the closest points on the obstacle boundaries partition the intersection circle into *sector*s. We can see that each sector is the "pie shaped area" within the intersection circle as seen in Fig. 2.
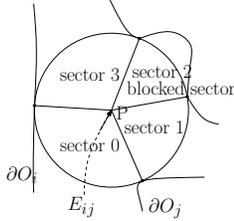

Fig. 2. The position of a vehicle is at the intersection. The *sector i* is the sector adjacent to the sector $i - 1$ in the counterclockwise direction.

Suppose that the vehicle under control moves along E_{ij} until it visits an intersection P, as illustrated on Fig.2. It will detect two closest points on ∂O_i and ∂O_j, since $P \in E_{ij}$. The sector that has these two closest points as its end points is defined as *sector* 0 for the intersection P. Intuitively, sector 0 is the sector through which the vehicle moves to reach the intersection P. It serves as a starting point for indexing the rest of the sectors. Suppose that there are n sectors in the intersection circle as seen on Fig.2. Looking into the page, we then index the sectors in the counterclockwise direction from sector 0. The index k satisfies $0 \leq k \leq n - 1$. When the vehicle leaves for the next intersection, it must move through another sector that contains the path leading to the next intersection. We call this sector as the *pointer sector*.

When two end points of a particular sector are on the same obstacle, the sector is called a *blocked sector* that is

illustrated as "sector 2" in Fig.2. An *open sector*, illustrated as "sector 1" and "sector 3" in Fig. 2, denotes a sector that is neither a blocked sector nor a sector 0.

If the intersection detected by a vehicle has an open sector that has not been visited by the vehicle, then the intersection is marked as *unexplored*. Otherwise, the intersection is marked as *explored*.

The following assumptions are made about the workspace and the vehicle's sensing and localization capability.

(A1) $\partial V(O_i)$ is a simple closed curve for all $O_i \in S_O$. In other words, $\partial V(O_i)$ is continuous and no self-intersection occurs.
(A2) there are finitely many intersections in W. All blocked sectors for these intersections are detectable by the vehicle.
(A3) $\bigcup_{O_i \in S_O} \overline{V}(O_i) = W$.
(A4) the initial position of the vehicle is that an obstacle other than O_M is detected to the right of the vehicle. The vehicle can distinguish O_M from other obstacles[5].

We call a closed loop that contains intersections connected by Voronoi edges an *enclosing boundary* if there is no unexplored intersection strictly inside such loop and the loop has no self-intersection. At any moment in BE algorithms, the enclosing boundary is unique.

B. Initialize the Enclosing Boundary

Algorithm 1 is to initialize the enclosing boundary. Suppose that the obstacle to the right of a vehicle is O_i. Under the tracking control law, the vehicle converges to the state that it moves along $\partial V(O_i)$ with ∂O_i to its right. We denote the first intersection on $\partial V(O_i)$ that the vehicle encounters as $P_{1,0}$. At each intersection that the vehicle encounters, it searches for an open sector in the counterclockwise direction, from the reader's view, from sector 0. Once an open sector is detected, then the vehicle moves through the sector. Iterating this, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed. The initial enclosing boundary B_0 is defined as the trajectory of a vehicle connecting this sequence of intersections until the vehicle is at $P_{1,0}$ for the second time.

C. Update the Enclosing Boundary

Algorithm 2 will expand B_0 to obtain B_k for $k = 1, 2, ...$ until B_k encloses all the obstacles except for O_M. We expand the enclosing boundary while maintaining it as a simple closed curve tracked by the vehicle in the clockwise direction.

The boundary expansion is guaranteed by two rules called the *sector selection rules* that decide which sector the vehicle should move through at an intersection and when to update the enclosing boundary.

Before stating the sector selection rules, we introduce a $(pointer + 1)$ sector which denotes a sector whose index is larger than the $(pointer)$ sector by one. The $(pointer)$ sector

---

[3]Proof of Theorem 1 is omitted in this paper.

[4]Suppose that the vehicle is at an intersection, then the closest points correspond to the points that have local minimal distances to the vehicle.

[5]Assumption (A4) is strictly speaking not a restriction, since the vehicle can initialize the heading orientation so that an obstacle other than O_M is detected to the right of the vehicle.
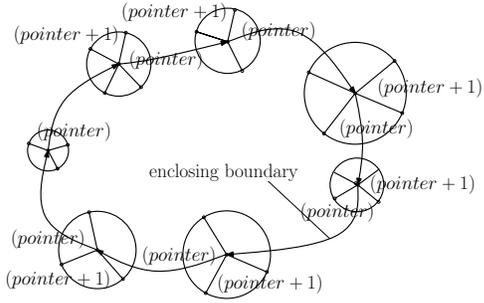
Fig. 3. The illustrative case to show $(pointer)$ sector and $(pointer + 1)$ sector stored at every intersection on the enclosing boundary.

and the $(pointer + 1)$ sector stored at every intersection on the enclosing boundary are illustrated on Fig.3.

The sector selection rules are stated below for two cases :

R1 When the vehicle visits an intersection on the enclosing boundary, the vehicle searches for an open sector in the counterclockwise direction from the $(pointer + 1)$ sector to sector 0. Once an open sector is detected, then the following condition is checked. If the vehicle would move through the open sector, then O_M would not lie to the right of the vehicle. If an open sector is detected that satisfies this condition, then the vehicle moves through the open sector. Otherwise, the vehicle moves through the $(pointer)$ sector.

R2 When the vehicle visits an intersection not on the enclosing boundary, the vehicle searches for an open sector in the counterclockwise direction from sector 0. Once an open sector is detected, then the vehicle moves through the open sector.

In the sector selection rule R1 and R2, the vehicle does not move through sector 0, since sector 0 is not an open sector according to definition. *Thus, using BE algorithms, the vehicle does not trace back along the Voronoi edge that the vehicle has just traversed.*

Any intersection in W is on $\partial V(O_i)$ for some $O_i \in S_O$. Thus, at any intersection, there exist two sectors that lead the vehicle to follow $\partial V(O_i)$ in the clockwise or in the counterclockwise direction. Therefore, we can always find an open sector that satisfies the sector selection rule R2.

Under the sector selection rules, the behavior of the vehicle is as follows. The vehicle moves along the enclosing boundary until it visits an intersection where there is an open sector that leads outside the enclosing boundary but will not force the vehicle to track O_M to its right. Then the vehicle marks the intersection as *head* and moves through the open sector. A singly linked list CS is initiated with the head. Thereafter the vehicle keeps moving and chooses sectors using the rule R2, inserting all intersections it encounters into CS. This process ends when the vehicle encounters the enclosing boundary again at an intersection. The vehicle marks this intersection as *tail* and insert tail into CS. We call the trajectory of a vehicle from the head to the tail as the *candidate segment*. After the vehicle gets to the tail, it uses rule R1 to determine which sector to move through.

The third rule is called the *boundary updating rule*. This rule regulates when to replace a segment of the current enclosing boundary with the candidate segment CS. The rule is as follows :

R3 If there is no unexplored intersection, strictly between the head and the tail, along the segment of enclosing boundary in the clockwise direction, then we replace the segment of enclosing boundary from the head to the tail by the candidate segment.
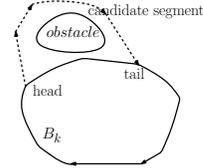

Fig. 4. The illustrative case where we update the enclosing boundary.

Suppose the current enclosing boundary is B_k that is the enclosing boundary updated after k steps. Fig. 4 illustrates the case where the boundary updating rule is satisfied. In this case, we update B_k by replacing the segment of enclosing boundary that starts from the head and ends at the tail by the candidate segment.

Fig.5 shows the case where the boundary updating rule is not satisfied. The dotted line indicates unexplored Voronoi edge. There are two unexplored intersections along the segment of enclosing boundary from the head to the tail. If the rule for updating B_k is not satisfied as illustrated in Fig.5, then we keep the enclosing boundary unchanged. To prevent the vehicle from repeatedly traversing the candidate segment that does not lead to boundary updates, the head of such candidate segment is recorded as a *disabled intersection* in a set D_k that is associated with B_k. If the vehicle encounters a disabled intersection, it will move along B_k to the next intersection.
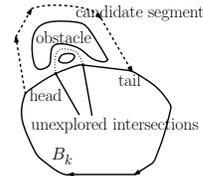

Fig. 5. The illustrative case where the boundary updating rule is not satisfied.

V. PROOF OF CONVERGENCE FOR BE ALGORITHMS

In this section, we prove the convergence of BE algorithms, i.e., both Algorithm 1 and Algorithm 2.

Lemma 1: Consider a vehicle and W satisfying assumptions (A1)-(A4). Suppose that the obstacle to the right of the vehicle is O_i. Then, using Algorithm 1, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed. Algorithm 1 terminates when the vehicle returns to the first intersection in the sequence.

Proof: Suppose that the obstacle to the right of the vehicle is O_i. Under the control law, the vehicle converges to the state that it moves along $\partial V(O_i)$ with ∂O_i to its right. We denote the first intersection on $\partial V(O_i)$ that the vehicle encounters as $P_{1,0}$, and label the intersections the vehicle will encounter if it follows $\partial V(O_i)$ with ∂O_i to its right as $(P_{1,0}, P_{2,0}, ..., P_{n,0})$. We omit the detailed proof of this lemma, but note that it should be organized with two steps :

1) Show that the vehicle moves to $P_{2,0}$.
2) Show that the vehicle visits $P_{2,0} \rightarrow P_{3,0}... \rightarrow P_{n,0} \rightarrow P_{1,0}$.

∎

To state Theorem 2, we need to introduce the following concepts : Let Q denote an obstacle, other than O_M, outside B_k such that $B_k \bigcap V(Q) \neq \emptyset$. If Q is such that $B_k \bigcap V(Q)$ is a connected line segment of B_k, then we call it an *addable obstacle Q^k*. Other than this possibility, there are two more possibilities that Q can have. Let Q^t denote an obstacle that $B_k \bigcap V(Q^t)$ is an intersection. Q^d denotes an obstacle such that $B_k \bigcap V(Q^d)$ is composed of disjoint line segments or intersections of B_k. $V(Q^t), V(Q^d)$, and $V(Q^k)$ are illustrated in Fig.6.
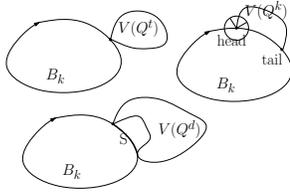

Fig. 6.   Illustration of $V(Q^t), V(Q^d), V(Q^k)$, and S.

Theorem 2: Consider a vehicle and W satisfying assumptions (A1)-(A4). The vehicle explores W using Algorithm 2. As long as there exists an obstacle other than O_M outside B_k, the following assertions hold :

1) B_k is a simple closed curve in the clockwise direction, and there is no unexplored intersection strictly inside B_k.
2) There exists an addable obstacle Q^k such that the vehicle moves along a path $CS \subset \partial V(Q^k)$ but $CS \neq \partial V(Q^k) \bigcap B_k$. The path intersects B_k at two intersections marked as head and tail. Further, CS is the candidate segment satisfying the rule for updating B_k.
3) After B_k is updated, the obstacle Q^k is inside the enclosing boundary.

Proof: Using Algorithm 1, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed according to Lemma 1. Therefore B_0 is in the clockwise direction from the reader's viewpoint, which is identical to $\partial V(O_i)$. Here, $B_0 = \partial V(O_i)$ is a simple closed curve using assumption (A1). Furthermore, no intersection is strictly inside B_0.

We prove by induction. Suppose that B_k is a simple closed curve in the clockwise direction and that there exists an obstacle other than O_M outside B_k. Also, suppose that there is no unexplored intersection strictly inside B_k. As long as there exists an obstacle other than O_M outside B_k, the following assertions hold :

1) show that there exists an addable obstacle Q^k as long as there exists an obstacle other than O_M outside B_k.
2) show that there exists no unexplored intersection strictly between the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$.
3) show that the vehicle moves along the path $CS \in \partial V(Q^k)$ and that the path intersects B_k at the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$.
4) show that, after new enclosing boundary B_{k+1} is generated, Q^k is inside B_{k+1}. B_{k+1} is a simple closed curve followed by the vehicle in the clockwise direction. There is no unexplored intersection strictly inside B_{k+1}.

1. First, we show that there exists Q as long as there is an obstacle other than O_M outside B_k. Suppose that all obstacles O_i outside B_k are such that $B_k \bigcap V(O_i) = \emptyset$. Then, since $\partial V(O_M)$ should enclose both B_k and O_i, $\partial V(O_M)$ has self-intersection.

Next, we prove the existence of Q^k by contradiction. Suppose all Q are either Q^d or Q^t. We first argue that Q^d must exist. If only Q^t exists, then $\partial V(O_M)$ has self-intersection, since $\partial V(O_M)$ should enclose both B_k and Q^t. For Q^d, call the disjoint boundary segments as $B_k \bigcap V(Q^d)$. Along B_k, there exists more than one line segment of B_k that connects these disjoint boundary segments. We select one segment S such that S and some edges of $\partial V(Q^d)$ form a closed loop that does not enclose Q^d. This closed loop is a simple closed curve, since self-intersection does not occur along $\partial V(Q^d)$ and $S \subset B_k$. S is illustrated on Fig.6. Inside the closed loop for Q_i^d, we iteratively find some other Q_{i+1}^d until no such Q_{i+1}^d exists. Voronoi edges in S that belongs to the inner most loop can not belong to $V(Q^t)$ for any Q^t, which implies that there exists addable obstacle Q^k inside the closed loop for Q_i^d. Therefore, by contradiction, there exists addable obstacle Q^k as long as there exists an obstacle other than O_M outside B_k.

2. We prove by contradiction. Suppose that an unexplored intersection exists strictly between the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$, then there exists an unvisited Voronoi edge meeting the unexplored intersection. Since we suppose that no unexplored intersection is strictly inside B_k, this unvisited Voronoi edge lies toward Q^k as illustrated on Fig.7. Hence, at this unexplored intersection, three edges of $\partial V(Q^k)$ meet, resulting in self-intersection of $\partial V(Q^k)$. This is a contradiction to assumption (A1).
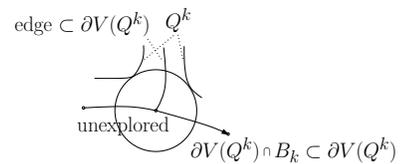

Fig. 7.   Three edges of $\partial V(Q^k)$ meet at an unexplored intersection on $\partial V(Q^k) \bigcap B_k$.

3-4. Proofs of step 3 and step 4 are omitted in this paper. ∎

Corollary 1: Under Algorithm 2, the enclosing boundary converges in finite time to the state that there is no obstacle other than O_M outside the enclosing boundary.

Proof: As long as there is an obstacle other than O_M outside B_k, we can generate B_{k+1} using Theorem 2. The process ends when there is no obstacle other than O_M outside B_k. Since there are finite number of obstacles, the process terminates in finite time. ∎

Corollary 2: When Algorithm 2 terminates, a complete Voronoi diagram is constructed for W [6].

VI. MATLAB SIMULATION RESULTS

We implement BE algorithms with the feedback control law (11) in MATLAB simulation. Fig. 8 shows a vehicle constructing a Voronoi diagram in a rectangular shaped workspace. The obstacle boundary is shown in red, and the segments of obstacle boundary detected by the range sensors are shown in blue. The trajectory of a vehicle is plotted as a green curve. On the vehicle's trajectory, intersections are marked with large yellow dots. We observe that all intersections are completely explored by the vehicle.
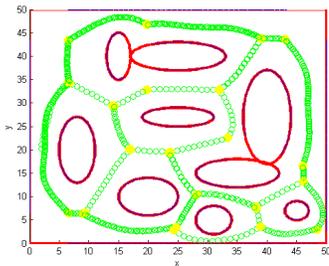

Fig. 8. We implement BE algorithms with the feedback control law (11) in MATLAB simulation.

VII. CONCLUSIONS

In this paper, we develop a provably convergent control law that enables a vehicle to follow Voronoi edges using range sensors. We then develop BE algorithms so that the Voronoi diagram structure of an unknown area can be constructed in finite time. BE algorithms implement decisions based on information gathered at each intersection that the vehicle encounters. We prove that such local decisions result in a global behavior that leads to the construction of a complete Voronoi diagram in finite time.

VIII. ACKNOWLEDGEMENTS

[6] Due to space limitations, proof of Corollary 2 is omitted in this paper.

REFERENCES

[1] S. J. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153 – 174, 1987.

[2] R. Klein, "Abstract Voronoi diagrams and their applications," *Computational Geometry and its Applications*, vol. 333, pp. 148 – 157, 1988.

[3] ——, *Concrete and Abstract Voronoi diagrams*. Springer, 1990.

[4] N. Sudha, S. Nandi, and K. Sridharan, "A parallel algorithm to construct Voronoi diagram and its VLSI architecture," in *Proc. of IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, 1999, pp. 1683–1688.

[5] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[6] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27(4), pp. 75 – 88, 2007.

[7] S. Bandyopadhyay and E. J. Coyle, "Minimizing communication costs in hierarchically clustered networks of wireless sensors," *Wireless Communications and Networking*, vol. 2, pp. 1274 – 1279, 2003.

[8] N. Rao, N. Stoltzfus, and S. Iyengar, "A retraction method for learned navigation in unknown terrains for a circular robot," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 699–707, 1991.

[9] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning - using the Voronoi diagram for a clearance-based shortest path," *IEEE Robotics and Automation Magazine*, vol. 15, pp. 58–66, 2008.

[10] S. Ahn, N. L. Doh, W. K. Chung, and S. Y. Nam, "The robust construction of a generalized Voronoi graph (GVG) using partial range data for guide robots," *Industrial Robot: An International Journal*, vol. 35, pp. 259 – 265, 2008.

[11] S. Garrido, L. Moreno, D. Blanco, and F. Martin, "Exploratory navigation based on Voronoi transform and fast marching," in *IEEE International Symposium on Intelligent Signal Processing*, Xiamen, China, 2007, pp. 1–6.

[12] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized Voronoi diagram," *The International Journal of Robotics Research*, vol. 19, pp. 96–125, 2000.

[13] H. Choset, I. Konukseven, and J. Burdick, "Mobile robot navigation: issues in implementation the generalized Voronoi graph in the plane," in *Proc. of IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Washington DC, USA, 1996, pp. 241 – 248.

[14] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 125 – 137, 2001.

[15] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Incremental construction of the hierarchical generalized Voronoi graph," *The International Journal of Robotics Research*, vol. 19, pp. 126–148, 2000.

[16] K. Nagatani and H. Choset, "Toward robust sensor based exploration by constructing reduced generalized Voronoi graph," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Korea, 1999, pp. 1687–1692.

[17] P. Svec, "Using methods of computational geometry in robotics," Ph.D. dissertation, Brno University of Technology, 2007.

[18] F. Zhang, E. Justh, and P. S. Krishnaprasad, "Boundary following using gyroscopic control," in *Proc. of 43rd IEEE Conf. on Decision and Control*, Atlantis, Paradise Island, Bahamas, 2004, pp. 5204–5209.

[19] F. Zhang, A. O'Connor, D. Luebke, and P. S. Krishnaprasad, "Experimental study of curvature-based control laws for obstacle avoidance," in *Proc. of IEEE International Conf. on Robotics and Automation*, New Orleans, LA, USA, 2004, pp. 3849–3854.

[20] F. Zhang and N. E. Leonard, "Coordinated patterns of unit speed particles on a closed curve," *Systems and Control Letters*, vol. 56, pp. 397–407, 2007.

[21] F. Zhang, D. M. Fratantoni, D. Paley, J. Lund, and N. E. Leonard, "Control of coordinated patterns for ocean sampling," *International Journal of Control*, vol. 80, pp. 1186–1199, 2007.

[22] P. V. Reddy, E. W. Justh, and P. Krishnaprasad, "Motion camouflage in three dimensions," in *Proc. of 45th IEEE Conf. on Decision and Control*, San Diego, CA, USA, 2006, pp. 3327–3332.

[23] C. Samson, "Control of chained systems: Application to path-following and time-varying point-stabilization of mobile robots," *IEEE Transactions on Automatic Control*, vol. 40, pp. 64–77, 1995.