Autonomous Robots manuscript No.
(will be inserted by the editor)

A provably complete exploration strategy by constructing Voronoi diagrams

Jonghoek Kim · Fumin Zhang · Magnus Egerstedt

Received: date / Accepted: date

Abstract We present novel exploration algorithms and a control law that enables the construction of Voronoi diagrams over unknown areas using a single vehicle equipped with range sensors. The control law uses range measurements to make the vehicle track Voronoi edges between obstacles. The exploration algorithms make decisions at vertices in the Voronoi diagram to expand the explored area until a complete Voronoi diagram is constructed in finite time. Our exploration algorithms are provably complete, and the convergence of the control law is guaranteed. Simulations and experimental results are provided to demonstrate the effectiveness of both the control law and the exploration algorithms.

Keywords Voronoi Diagrams · Map-making Algorithms · Robot Control

1 Introduction

This paper addresses the problem of exploring an unknown workspace using one vehicle equipped with range sensors. Such sensors have ability to determine a point on obstacle boundary that is closest to the vehicle. We call such a point the *closest point*. If boundary curves appear on both the left and the right hand sides of the vehicle, then a closest point can be determined on each boundary. The path that has equal distances from these closest points is a 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 under certain algorithms, then we consider the workspace as being completely explored.

Voronoi diagrams have been widely used in various areas such as biology [6,24], computational geometry [15,19,20,28], VLSI design [32], and sensor networks [3,12,23]. In robotics, Voronoi diagrams have been utilized to obtain paths that satisfy minimum clearance requirements [1, 4, 16, 29, 34]. Voronoi diagrams can be generalized into higher dimensions, and also fit a wide class of robots with higher dimensional configuration spaces [7, 8]. 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)[1] in [7, 10, 11, 26]. The exploration of an unknown workspace by incrementally constructing the Voronoi diagram was previously achieved in [8, 29, 33]. But convergence of the algorithms is not proved in the above references.

In order for the vehicle to follow a Voronoi edge, we develop a Voronoi edge tracking control law that is provably convergent. This law is based on the shape dynamics derived in [36] and [38], where a gyroscopic feedback control law was developed to control the interaction between the vehicle and the closest point so that the vehicle follows the obstacle boundary either to the left or to the right. This controller design method was generalized to cooperative motion patterns on closed curves for multiple vehicles in [35, 37], and extended to the design of pursuit-evasion laws in three dimensions [30]. The closest point was also used for path following in earlier works such as [31]. Our curve tracking control law extends previous works by using informa-

Jonghoek Kim · Fumin Zhang · Magnus Egerstedt
School of Electrical and Computer Engineering,
Georgia Institute of Technology, USA
E-mail: jkim37@gatech.edu
fumin@gatech.edu
magnus@ece.gatech.edu

[1] Our notion of Voronoi diagrams is similar to the RGVG in that no Voronoi edge is connected to the obstacle boundaries.

tion from the closest points on both sides of the vehicle. This results in a tracking behavior of the Voronoi edge between two obstacles.

To make a robot track a Voronoi edge, the authors of [8] used a numerical continuation method that relies on two iterative stages : a prediction step and a correction procedure. Since the prediction step makes the robot move off the GVG edge, this continuation method may produce zigzag movements of the robot [9]. Furthermore, after the correction procedure, the robot must rotate again to re-orient itself on a Voronoi edge. These rotations may take time and cause additional wheel slippage [9]. Later, a control law was presented in [9], which provided a smoother path than the numerical continuation method [8]. However, smoothness of the robot's trajectory was not ensured. Note that the curvature of obstacle boundaries was not considered in [8,9]. In contrast, our tracking control law utilizes estimated curvature that is well defined along the trajectory of the vehicle. Hence, we can guarantee that the trajectory of the vehicle is smooth. MATLAB simulations in Section 7.1 verifies that our control law produces a smooth path.

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

The BE algorithms are composed of two algorithms, denoted by Algorithm 1 and Algorithm 2 in this paper. 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 such 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.

In the BE algorithms, only the graph structure representing the boundary of the explored area is maintained and updated based on two simple rules. We do not explicitly search for the shortest path in the graph as in many other exploration algorithms [10, 29, 33]. Hence, the BE algorithms may have lower computational load.

We implement the algorithms and the control law on a miniature robot localizing itself based on an odometry system. The robot uses only Infrared (IR) sensors for range measurements. Both simulations and experimental results demonstrate the effectiveness of the algorithms.

The paper is organized as follows. Section 2 presents the workspace to be explored. Section 3 introduces the provably convergent control law to track Voronoi edges. Section 4 discusses the BE algorithms. Section 5 provides proofs for the convergence of the BE algorithms. Section 6 analyzes the efficiency of the BE algorithms. Section 7 demonstrates simulation and experimental results, and Section 8 provides conclusions.

2 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 $O_i \subset W$. 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 $S_O = \{O_1, O_2, ..., O_M\}$.

Obeying the conventions established in the literature on Voronoi diagrams [2,7,20,21,26], we define the *Voronoi cell* for an obstacle O_i as the set of points that is closer to O_i than to any other obstacle in S_o for $i = 1, 2, ..., M$ i.e.

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

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 [20] 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

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

3 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. We assume that the range sensors of the vehicle can detect two obstacle boundaries on the right and the left hand side of the vehicle. Then, on each obstacle boundary, the vehicle can determine a point that is closest to the vehicle. The feedback control law uses measurements at the two closest points on the right and the left hand sides of the vehicle.

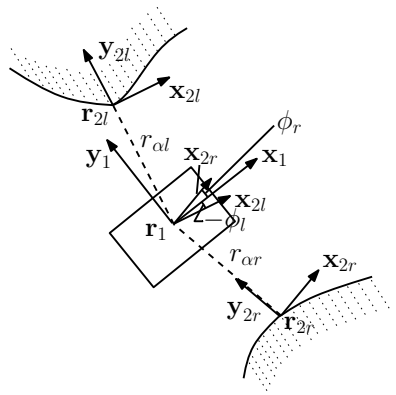

Fig. 1 A vehicle with boundary curves to the left and to the right of the vehicle.

We first introduce the shape dynamics that govern the relationship between the vehicle and the closest points. We then derive the tracking control law, and prove its convergence.

3.1 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 $\mathbf{r}_{2r}$. ϕ_r is the angle measured counter-clockwise from $\mathbf{x}_1$ to $\mathbf{x}_{2r}$, the tangent vector at $\mathbf{r}_{2r}$. 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 we define $r_{\alpha r} = \|\mathbf{r}_{\alpha r}\|$.

The quantities $\mathbf{r}_{2l}$, $\mathbf{x}_{2l}$, ϕ_l, $\mathbf{r}_{\alpha l}$, and $r_{\alpha l}$ are defined to the left of the vehicle in the same fashion as those to the right of the vehicle.

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 $-\pi/2 < \phi_r < \pi/2$.

Considering the boundary curve to the right of the vehicle, the shape dynamics are given by [36] 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.

3.2 Tracking Control and Convergence Analysis

In this section, we design the tracking control law based on a 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 that balances the control for alignment and the control for vehicle position. 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 a Voronoi edge. The time derivative of V is

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

where we have used the 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 $\dot{V} \leq 0$. One choice of u that leads to $\dot{V} \leq 0$ is

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

where $\mu > 0$ is a constant gain for the tracking controller. According to [36], we see that the steering control u corresponds to the curvature of the vehicle's trajectory at the moment when u is applied. Hence, as long as the control law (11) is not singular (denominator of (11) is not zero), curvature is well defined along the trajectory of the vehicle. Hence, we can guarantee that the trajectory of the vehicle is smooth.

The time derivative of V in (10) with u given by (11) is

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

By letting $\phi = \frac{\phi_l+\phi_r}{2}$, we get $\dot{V} = -\mu\frac{\sin^2(\phi)}{\cos(\phi)}$. In addition, $-\pi/2 < \phi < \pi/2$ is derived using (4). Within this range, we obtain $-\infty < \dot{V} \leq 0$ which is unbounded. $\dot{V} = 0$ if and only if $\phi = 0$, since $-\pi/2 < \phi < \pi/2$. As $|\phi|$ increases from 0 to $\pi/2$, $\dot{V}$ monotonically decreases to $-\infty$. This is to penalize misalignment between the heading direction of the vehicle and the tangent vector to the Voronoi edge.

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 smooth Voronoi edge.*

Proof For each trajectory that initially satisfies (4), there exists a compact sublevel set Ω of V such that the trajectory remains in Ω for all future time. Then, by LaSalle's Invariance Principle [17], the trajectory converges to the largest invariant set I within the set E that contains all points in Ω where $\dot{V}=0$. The set E in this case is the set of all points in Ω such that $\phi_l+\phi_r=0$. Thus, at any point in E, the dynamics are expressed as

$$E=\{(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r) | \phi_l+\phi_r=0\}. \tag{13}$$

Since the trajectory converges to the largest invariant set I within the set E, we obtain $\dot{\phi}_l+\dot{\phi}_r=0$ in I. Therefore, using (6) and (8), we derive

$$(\frac{\kappa_r}{1-\kappa_r r_{\alpha r}})\cos(\phi_r)+(\frac{\kappa_l}{1+\kappa_l r_{\alpha l}})\cos(\phi_l)-2u=0. \tag{14}$$

Applying (11), we get

$$2\lambda(r_{\alpha l}-r_{\alpha r})(\cos(\phi_l)+\cos(\phi_r))+\mu\sin(\frac{\phi_l+\phi_r}{2})=0. \tag{15}$$

In order to satisfy (15), $r_{\alpha l}-r_{\alpha r}=0$ is required, since $\phi_l+\phi_r=0$ inside the set E. Therefore, the largest invariant set I is expressed as

$$I=\{(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r) | r_{\alpha l}=r_{\alpha r}, \phi_l+\phi_r=0\}. \tag{16}$$

Thus, we can conclude that $(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r)$ converges to the equilibrium where $r_{\alpha l}=r_{\alpha r}$ and $\phi_l=-\phi_r$. If the vehicle is equidistant from two closest points on the obstacle boundaries and the heading direction of the vehicle is aligned to the tangent vector to the Voronoi edge, then the vehicle moves along the Voronoi edge. This implies that, as the vehicle converges to the state I in (16), it converges to move along the Voronoi edge. □

By means of the LaSalle's Invariance Principle, we can conclude asymptotic convergence. This may cause a problem for a vehicle to track a Voronoi edge with finite length. This problem can be alleviated by noticing that the convergence rate of the control law depends on the controller gain μ (see (12)). Larger gain μ and weight λ will enable the vehicle converges to a Voronoi edge faster, which has been confirmed by rigorously compute the eigenvalues of the Jacobian matrix for linearized closed loop dynamics near the tracking equilibrium. If a lower bound of the length of Voronoi edges within the workspace is known, then μ can be selected so that the vehicle gets sufficiently close to Voronoi edges in finite time. Such a lower bound can be estimated based on the length of the Voronoi edges already detected by the vehicle, which will result in an adaptive gain μ. The details of the gain adjustment algorithm is not the main focus of this paper.

4 The Boundary Expansion (BE) Algorithms

In this section, we propose the boundary expansion (BE) algorithms that enable the vehicle to construct the Voronoi diagram of W by traversing all Voronoi edges E_{ij} for $i,j=1,2,...,M$.

4.1 Definitions and Assumptions

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

- there exists a circle centered at P intersecting obstacle boundaries at more than two points. These points on obstacle boundaries are called the *closest points* at the intersection. If the vehicle is at an intersection, then the closest points correspond to the points that have local minimal distances to the vehicle.
- the interior of the circle does not intersect any obstacles. The circle is called an *intersection circle* (see 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 (see Fig. 2).

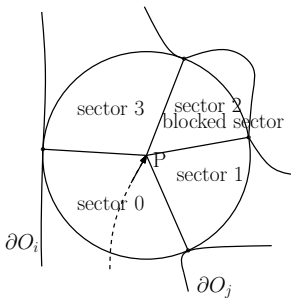

Fig. 2 The position of the vehicle is at the intersection P. The circle is the intersection circle. The closest points partition the circle into sectors. The *sector* i is the sector adjacent to the sector $i-1$ in the counter-clockwise direction.

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 counter-clockwise direction from sector 0. The index k satisfies $0 \leq k \leq n-1$.

When two end points of a particular sector are on the same obstacle, the sector is called a *blocked sector*, which 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 the 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 each $O_i \in S_O$. 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[2].

(A3) $\bigcup_{O_i \in S_O} \overline{V}(O_i) = W$.

(A4) the initial position of the vehicle is such that an obstacle other than O_M is detected to the right of the vehicle[3]. The vehicle can distinguish O_M from other obstacles[4].

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

4.2 Data Structures

The data structures used in the BE algorithms are summarized in Table 1. For each intersection detected by the vehicle, we store the coordinates of the intersection. The enclosing boundary can then be represented by a circularly linked list L constructed by linking the intersections.

Table 1 Table of Data Structures and Operations

L_u : singly linked list representing the enclosing boundary under construction.
L_u.Insert(P) : insert an intersection P at the end of the linked list L_u.
L : circularly linked list representing the current enclosing boundary.
HT=L.seg(*head*,*tail*) : segment of L that starts from the *head* and ends at the *tail*.
L_r= L.Remove(HT) : remove the segment HT from L resulting in L_r.
CS : singly linked list representing the candidate segment.
L=L_r.Combine(CS) : combine the linked list L_r with CS resulting in updated L.
G : graph structure, representing the Voronoi diagram under construction, which contains a list of intersections and an adjacency matrix.
G.Update(P) : update entries of the adjacency matrix associated to an intersection P.
G.Expand(P) : expand the adjacency matrix to include an intersection P, and update entries of the matrix associated to P.
HT.Search(*unexplored*) : search for unexplored intersections in the linked list HT. If there is no unexplored intersection, return $NULL$.
D_k : disabled intersection set of B_k (the enclosing boundary updated after k steps).
D_k.Store(P) : store an intersection P in D_k.

We use a graph structure G to represent the Voronoi diagram under construction. G contains a list of distinct intersections together with an adjacency matrix whose entries indicate whether a particular edge is in the graph. When the vehicle detects an intersection P that has not been stored in G, then the adjacency matrix is expanded to include P.

4.3 Initialize the Enclosing Boundary

Algorithm 1 is to initialize the enclosing boundary. Suppose that the obstacle to the right of the 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 counter-clockwise direction, from the reader's view, from sector 0. Once an open sector is detected, 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 sequence of Voronoi edges connecting this

[2] The experiments in Section 7 verify that the robot can detect a blocked sector using IR sensors.

[3] 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.

[4] We can consider specific sensors deployed along O_M so that the vehicle can distinguish O_M from other obstacles. Or, O_M may have a different shape (or color) from other obstacles.

intersection sequence until the vehicle is at $P_{1,0}$ for the second time.

Algorithm 1 Construct the Initial Enclosing Boundary

$i \leftarrow 1$.
repeat
  The vehicle encounters an intersection.
  $P_{i,0} \leftarrow$ the intersection.
  Search for an open sector in the counter-clockwise direction from sector 0. The vehicle moves through the first open sector.
  $P_{i,0}$.pointer $\leftarrow$ first open sector.

  if $P_{i,0}$ has an open sector that has not been visited by the vehicle **then**
    $P_{i,0}$.mark $\leftarrow$ *unexplored*.
  else
    $P_{i,0}$.mark $\leftarrow$ *explored*.
  end if
  L_u.Insert($P_{i,0}$).
  if $P_{i,0} \in G$ **then**
    G.Update($P_{i,0}$).
  else
    G.Expand($P_{i,0}$).
  end if
  $i \leftarrow i + 1$.
until the vehicle encounters $P_{1,0}$ for the second time.
$L \leftarrow L_u$.

4.4 Update the Enclosing Boundary

Let B_k denote the enclosing boundary updated after k steps. 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 the *(pointer) sector* and the *(pointer + 1) sector*. When the vehicle on the enclosing boundary leaves for the next intersection along the enclosing boundary in the clockwise direction, it must move through another sector that contains the path leading to the next intersection. We call this sector the (*pointer*) sector. The (*pointer* + 1) sector denotes a sector whose index is larger than the (*pointer*) sector by one. The (*pointer*) sector and the (*pointer* + 1) sector stored at every intersection on the enclosing boundary are illustrated on Fig. 3.

The sector selection rules are stated for two cases :

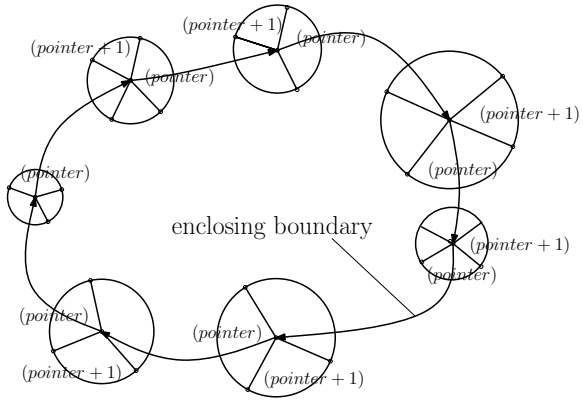

Fig. 3 The illustrative case to show the (*pointer*) sector and the (*pointer* + 1) sector stored at every intersection on the enclosing boundary.

R1 When the vehicle visits an intersection on the enclosing boundary, the vehicle searches for an open sector in the counter-clockwise direction from the (*pointer* + 1) sector to sector 0. Once an open sector is detected, 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, 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 counter-clockwise direction from sector 0. Once an open sector is detected, the vehicle moves through the open sector.

Suppose that the vehicle is on the Voronoi edge $E_{ij} \subset \partial V(O_i)$, where $i \neq j$. At any intersection on $\partial V(O_i)$, there exist two sectors that lead the vehicle to follow $\partial V(O_i)$ in the clockwise or in the counter-clockwise direction. Therefore, the vehicle 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 inserts *tail* into CS. We call

the trajectory of the vehicle from the *head* to the *tail* the *candidate segment*. After the vehicle gets to the *tail*, it uses the rule R1 to determine which sector to move through.

We introduce 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.

Suppose the current enclosing boundary is B_k. Figure 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. Figure 5 shows the update of L, the data structure of the enclosing boundary, when the boundary updating rule is satisfied.

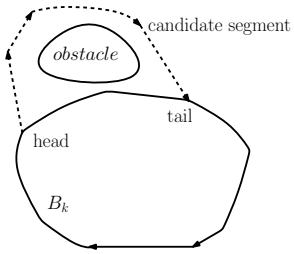

Fig. 4 The illustrative case where we update the enclosing boundary.

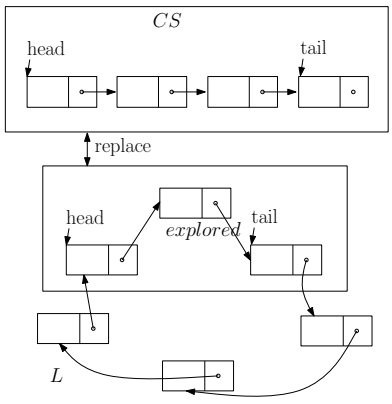

Fig. 5 Update of the enclosing boundary when the boundary updating rule is satisfied.

Figure 6 shows one case where the boundary updating rule is not satisfied. The dotted curve indicates the 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. 6, 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 a 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 ignore this intersection and move along B_k to the next intersection.

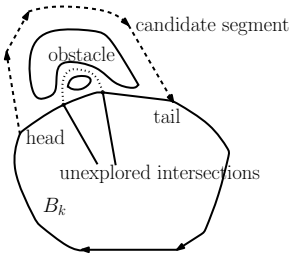

Fig. 6 The illustrative case where the boundary updating rule is not satisfied.

5 Convergence of the BE algorithms

In this section, we prove the convergence of the boundary expansion 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 organize our proofs in two steps:

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

1. For convenience, we call the closest point on ∂O_i as $C_{\partial O_i}$. At $P_{1,0}$, $C_{\partial O_i}$ is to the right of the vehicle. Suppose that there are n sectors at $P_{1,0}$. Sector 0 and sector

Algorithm 2 Boundary Expansion

N denotes the number of intersections on the circularly linked list L. Label the intersections on L in the clockwise direction as $P_{1,0}$,$P_{2,0}$,...,$P_{N,0}$. $i \leftarrow 1$ and $k \leftarrow 0$.
while there is an obstacle other than O_M outside the enclosing boundary **do**
  The vehicle visits $P_{i,k}$ on L.
  if $P_{i,k} \notin D_k$ and there exists an open sector, satisfying the sector selection rule R1, outside the enclosing boundary **then**
    $m \leftarrow 1$. $E_1 \leftarrow P_{i,k}$. $MeetTail \leftarrow 0$.
    while $MeetTail \neq 1$ **do**
      The vehicle finds E_m.
      if $m == 1$ **then**
        Move through the open sector selected using the rule R1.
      else
        Search for an open sector satisfying the rule R2, and move through the selected open sector.
      end if
      E_m.pointer$\leftarrow$ selected sector.
      if E_m has an open sector that has not been visited by the vehicle **then**
        E_m.mark $\leftarrow$ *unexplored*.
      else
        E_m.mark$\leftarrow$ *explored*.
      end if
      CS.Insert(E_m).
      if $E_m \in G$ **then**
        G.Update(E_m).
      else
        G.Expand(E_m).
      end if
      if $m \neq 1$ and the vehicle intersects L **then**
        $n \leftarrow 1$.
        while $E_m \neq P_{n,k}$ **do**
          $n \leftarrow n + 1$.
        end while
        $T \leftarrow n$ and $MeetTail \leftarrow 1$.
      else
        $m \leftarrow m + 1$.
      end if
    end while
    $head \leftarrow E_1$ and $tail \leftarrow P_{T,k}$.
    HT=L.seg($head$,$tail$).
    if $HT \neq NULL$ and HT.Search($unexplored$)$\subset$ ($head$, $tail$) **then**
      L_r=L.Remove(HT).
      L=L_r.Combine(CS).
      N denotes the number of intersections on L.
      $P_{1,k+1} \leftarrow tail$. Relabel the intersections on L in the clockwise direction as $P_{1,k+1}$,$P_{2,k+1}$,...,$P_{N,k+1}$.
      $i \leftarrow 1$ and $k \leftarrow k + 1$.
    else
      D_k.Store($head$). $i \leftarrow T$.
    end if
  else
    $i \leftarrow i + 1$.
  end if
  if $i > N$ **then**
    $i \leftarrow i - N$.
  end if
end while

$n-1$ have the common closest point at $C_{\partial O_i}$. The vehicle moves through the sector $n-1$ if it is not blocked. If sector $n-1$ is blocked, then the sector selection rule R2 is applied so that the vehicle moves through the next open sector having $C_{\partial O_i}$ to the right of the vehicle. Therefore, the vehicle tracks $\partial V(O_i)$ with ∂O_i to its right and will encounter $P_{2,0}$.

2. Consider the case where the vehicle visits $P_{k,0}$ starting from $P_{k-1,0}$ for all $2 \leq k \leq n$. Similar to step 1, using the sector selection rule R2, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right until it visits $P_{k+1,0}$.

By induction, if the vehicle uses the sector selection rule R2 at $P_{1,0}, P_{2,0}, ...P_{n,0}$, then it visits the intersections following the sequence $P_{1,0} \rightarrow P_{2,0}... \rightarrow P_{n,0} \rightarrow P_{1,0}$. Algorithm 1 ends when the vehicle returns to $P_{1,0}$. □

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 edge 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 edge segments or intersections of B_k. $V(Q^t), V(Q^d)$, and $V(Q^k)$ are illustrated in Fig. 7.

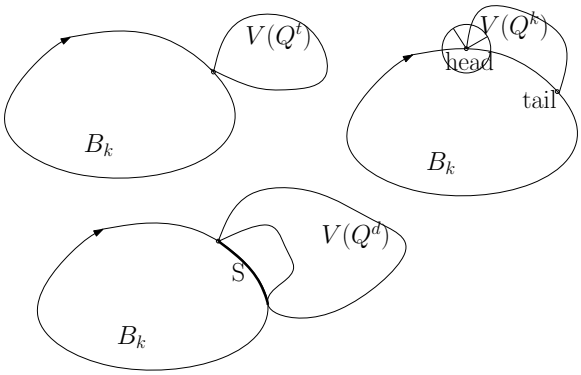

Fig. 7 Illustration of $V(Q^t), V(Q^d), V(Q^k)$, and S. Q^k is addable. However, Q^d and Q^t are not addable.

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 traversed 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 that can be marked as head and tail. Further, CS is the candidate segment satisfying the rule for updating B_k.*
3. *B_k is updated so that 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. Suppose that there is no unexplored intersection strictly inside B_k. Now, we organize our proofs in four steps:

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 intersection (*head*) and the ending intersection (*tail*) 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 traversed by the vehicle in the clockwise direction, and 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 exist one or more edge segments of B_k connecting 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. S is illustrated on Fig. 7. This closed loop can be constructed as a simple closed curve, since self-intersection does not occur along either $\partial V(Q^d)$ and $S \subset B_k$. If S intersects $\partial V(Q^d)$ at more than two points, we can always select a shorter segment of S so that a simple closed loop can be constructed. We call this closed loop Q_1^d. Inside the closed loop Q_i^d, we iteratively find another loop for Q_{i+1}^d until no more Q_{i+1}^d exists. Voronoi edges in S that exist along the inner most loop belong to neither $V(Q^d)$ nor $V(Q^t)$ for any Q^t, which implies that there exists an addable obstacle Q^k inside the inner most loop. Therefore, by contradiction, there exists an 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. 8. 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).

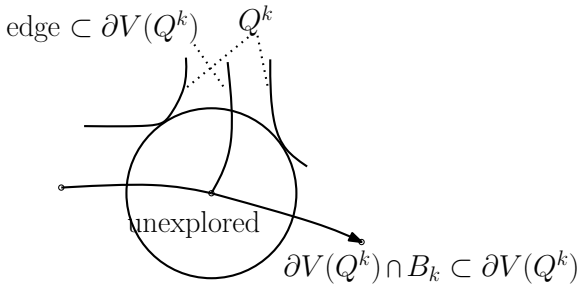

Fig. 8 Three edges of $\partial V(Q^k)$ meet at an unexplored intersection on $\partial V(Q^k) \bigcap B_k$.

3. We suppose that the vehicle has tracked B_k in the clockwise direction until it visits the starting intersection of $\partial V(Q^k) \bigcap B_k$. Then, we mark the starting intersection as *head* and mark the ending intersection of $\partial V(Q^k) \bigcap B_k$ as *tail*. Note that the direction of $\partial V(Q^k) \bigcap B_k$ is from the *head* to the *tail*, since B_k is in the clockwise direction.

When the vehicle visits the *head*, there exists an open sector outside the enclosing boundary, as illustrated on Fig. 7. Then, according to the rule R1, the vehicle moves through the open sector outside the enclosing boundary with Q^k to the vehicle's right. Thereafter, it chooses sectors using the rule R2 and moves along Voronoi edges.

We label the intersections the vehicle encounters if it follows $\partial V(Q^k)$ with Q^k to its right as ($E_1 = head, E_2$, ..., $E_n = tail$). Similar to the proof of Lemma 1, the vehicle starting from E_m moves along $\partial V(Q^k)$ with Q^k to its right until it visits E_{m+1}. The sequence of Voronoi

edges connecting the intersection sequence ($E_1 = head, E_2$, ..., $E_n = tail$) is defined as the candidate segment $CS \subset \partial V(Q^k)$.

4. The boundary updating rule for B_k is satisfied. Thus, we update B_k by substituting $\partial V(Q^k) \bigcap B_k$ for CS. There is no unexplored intersection strictly inside B_{k+1}, because there exists no unexplored intersection strictly between the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$.

We now prove that B_{k+1} is a simple closed curve in the clockwise direction. Since self-intersection of CS can not occur as we substitute $\partial V(Q^k) \bigcap B_k$ for CS, B_{k+1} is a simple closed curve. In addition, the direction of B_{k+1} is in the clockwise direction, since the direction of $\partial V(Q^k) \bigcap B_k$ is the same as that of CS.

Next, since $CS \bigcup (\partial V(Q^k) \bigcap B_k) \subset \partial V(Q^k)$ is a simple closed curve, $CS \bigcup (\partial V(Q^k) \bigcap B_k) = \partial V(Q^k)$. After we generate B_{k+1}, the area enclosed by $CS \bigcup (\partial V(Q^k) \bigcap B_k)$ is inside B_{k+1}, i.e., Q^k is inside B_{k+1}. We have proved all the statements in Theorem 2. □

Corollary 1 *Under Algorithms 1 and 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.*

Proof : When Algorithm 2 terminates, there is no unexplored intersection outside the final enclosing boundary, since there is only O_M outside. And, according to Theorem 2, there is no unexplored intersection strictly inside the enclosing boundary either. Thus, all the intersections in W are explored. This implies that all the Voronoi edges in W are visited by the vehicle. Consequently, the trajectory of the vehicle depicts all the Voronoi edges in W and a complete Voronoi diagram is constructed. □

6 Performance Analysis

In this section, we provide an analytical upper bound for the total time spent to construct Voronoi diagrams in a regularized workspace. Each obstacle, other than O_M, is now simplified as one site (called a generator in [13]). Then the workspace is partitioned by a centroidal Voronoi tessellation.

As the number of Voronoi cells in a bounded workspace increases, each Voronoi cell approaches to hexagonal shape [13,27]. Thus, we analyze the performance of the BE algorithms in the workspace where each cell has a hexagonal shape with identical size. Hexagonal Voronoi cells can be built using identical circular obstacles, as illustrated on Fig. 9.

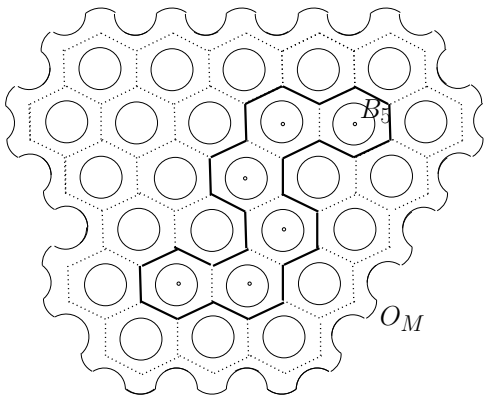

Fig. 9 All Voronoi cells, except for $V(O_M)$, have hexagonal shapes with identical size. Inside B_5, there are 6 obstacles.

Theorem 3 *Consider a single unit speed vehicle and workspace W with assumptions (A1)-(A4) satisfied. Suppose that there are M obstacles such that all obstacles, except for O_M, have hexagonal Voronoi cells with identical size. Using the BE algorithms, the exploration time is bounded above by $T(\frac{1}{2}M^2 - \frac{1}{2}M)$ where T denotes the time for a vehicle to traverse along the edges of one hexagonal Voronoi cell.*

Proof : Consider the time to build B_0. Since there is only one Voronoi cell inside B_0, the time to construct B_0 is

$$T_{B_0} = T. \tag{17}$$

Next, consider the time to generate B_{k+1} from B_k where $k \geq 0$. Suppose that B_k is generated and that the vehicle is at the *tail* of the candidate segment (CS) for generating B_k.

Using Theorem 2, at least one of the Voronoi cells, which is outside B_k and intersects the perimeter of B_k, is an addable obstacle Q^k. This addable obstacle Q^k will be an obstacle that is inside B_{k+1}. The vehicle moves along B_k to reach the starting intersection (*head*) of $\partial V(Q^k) \bigcap B_k$. The vehicle's maximal traversal distance to meet the starting intersection of $\partial V(Q^k) \bigcap B_k$ is bounded above by the length of B_k. Note that the vehicle has unit speed and that the number of Voronoi cells, which are inside B_k, is $k+1$. Therefore, the length of B_k is bounded above by $(k+1)T$. In addition, the

length of CS, which connects the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$, is bounded above by T, since the vehicle has unit speed. Hence, we derive

$$T_{B_{k+1}} \leq T_{B_k} + (k+1)T + T, \tag{18}$$

where T_{B_k} denotes the time for a vehicle to construct B_k. Using (18), we obtain

$$T_{B_k} \leq T(\tfrac{1}{2}k^2 + \tfrac{3}{2}k + 1), \tag{19}$$

since $T_{B_0} = T$ (see (17)). There are $k+1$ and $M-1$ obstacles inside B_k and $\partial V(O_M)$ respectively. Therefore, our algorithms terminate when

$$k + 1 = M - 1. \tag{20}$$

Hence, replacing $k+1$ in (19) by $M-1$, we obtain the time upper bound for the construction of Voronoi diagrams using the BE algorithms as

$$T_c \leq T(\frac{1}{2}M^2 - \frac{1}{2}M). \tag{21}$$

Therefore, the expected construction time is $O(M^2)$. □

7 Simulation and Experimental Results

We introduce two strategies to improve the time efficiency of the BE algorithms. Both the improved BE algorithms and the feedback control law (11) are implemented in MATLAB simulations. To compare our algorithms with [10], we perform MATLAB simulations of the exploration algorithms and the control law in [10]. We then present the experimental results on a Khepera III robot in an environment with three obstacles.

7.1 Simulation Results

Figure 10 depicts the MATLAB simulation results using the exploration algorithms and the control law in [10], and Fig. 11 depicts the results using the BE algorithms and the control law. In both Fig. 10 and Fig. 11, the initial position of the vehicle is $(2, 20)$, and the obstacle boundaries are shown in thick red curves. The trajectory of the vehicle is plotted with blue points. Along the vehicle's trajectory, the intersections are marked with large green dots.

In [10] and related works, the vehicle moves through a sector to check whether the sector is open or blocked. In other words, the vehicle moves through a blocked sector until it detects a blocking obstacle boundary. Our simulation results show that 63.4 time unit is spent to finish the exploration in Fig. 10.

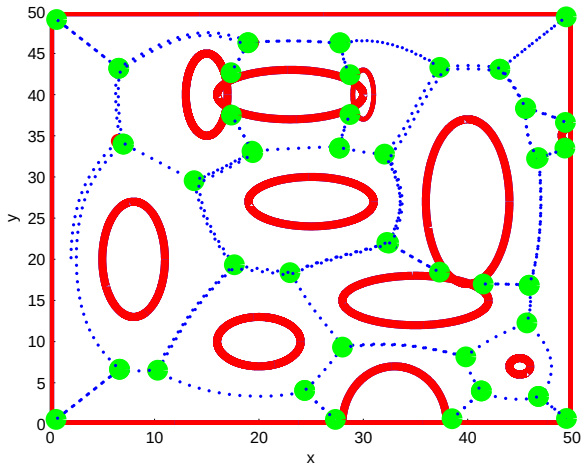

Fig. 10 The Voronoi diagram constructed by the vehicle using the exploration algorithms and the control law in [10].

The BE algorithms are theoretically sound, but we can improve the time efficiency of the algorithms without violating the correctness of the algorithms. We have implemented two strategies. The first strategy is inspired by the fact that the vehicle does not have to traverse the entire enclosing boundary to find an unexplored intersection. Whenever the vehicle finishes building a candidate segment, it plans the shortest path to reach the nearest unexplored intersection on the enclosing boundary. Once the vehicle reaches the unexplored intersection, it branches out of the loop to expand the enclosing boundary.

The second strategy is to store the candidate segment with a disabled intersection, as discussed in Section 4.4. If the boundary updating rule is not satisfied, we store the corresponding candidate segment as a *disabled candidate segment*. Whenever the enclosing boundary is updated, the vehicle checks the disabled candidate segment to see whether there is still an unexplored intersection from the *head* to the *tail*. If no unexplored intersection is found, then the disabled candidate segment will be enabled and boundary expansion can be performed using this candidate segment. This strategy updates the enclosing boundary without letting the vehicle traverse the disabled candidate segment again.

Figure 11 depicts the trajectory of the vehicle using the BE algorithms and the control law with improvement over time efficiency. The total exploration time is 36.3 time unit. The vehicle does not move through a blocked sector, since we assume that the range sensors of the vehicle can detect a blocked sector at an intersection. If this assumption is removed, and we allow the vehicle to detect a blocked sector by retracing behaviors (complete turning whenever the vehicle de-

tects a blocking obstacle boundary) as in [10], then the BE algorithms take 61.8 time unit to finish[5]. Hence, for the workspace illustrated in Fig. 10 and 11, the time efficiency of the improved BE algorithms is comparable to the algorithms in [10]. Even though more comparison may be necessary to formulate a definite conclusion on comparing the BE algorithms with the algorithms originated from [10], the difference in the behavior of the robot is significant enough to justify possible choices made in various contexts. The BE algorithms have added an option to the library of exploration algorithms.

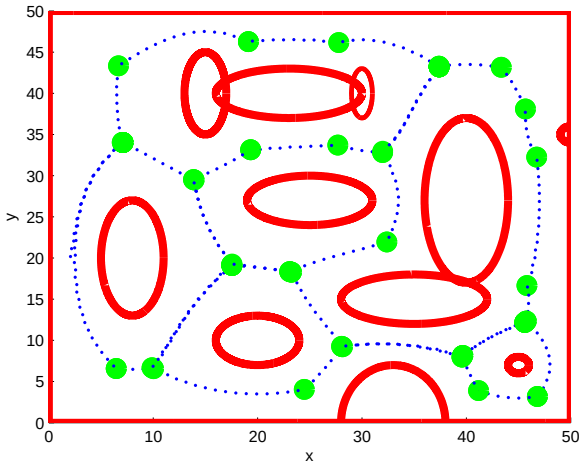

Fig. 11 The trajectory of the vehicle is built using the BE algorithms and the control law with improvement over time efficiency.

7.2 Experimental Results

The validity of our algorithms and the control law (11) is verified using a miniature robot Khepera III [25] that localizes itself based on an odometry system. The Khepera III robot has nine IR sensors and five sonar sensors. In the experiments, we use only IR sensors for range measurements.

As the robot maneuvers in the workspace, a MATLAB plot is displayed in real time to show the detected obstacle environment. Figure 12 shows the real time plot with corresponding obstacle environment. The Khepera III robot is depicted as a dotted circle. In addition, the trajectory of the robot is plotted as a blue curve. On the trajectory of the robot, intersections are marked with small circles.

[5] We omit the MATLAB figure for this case, since it is almost the same as Fig. 10.

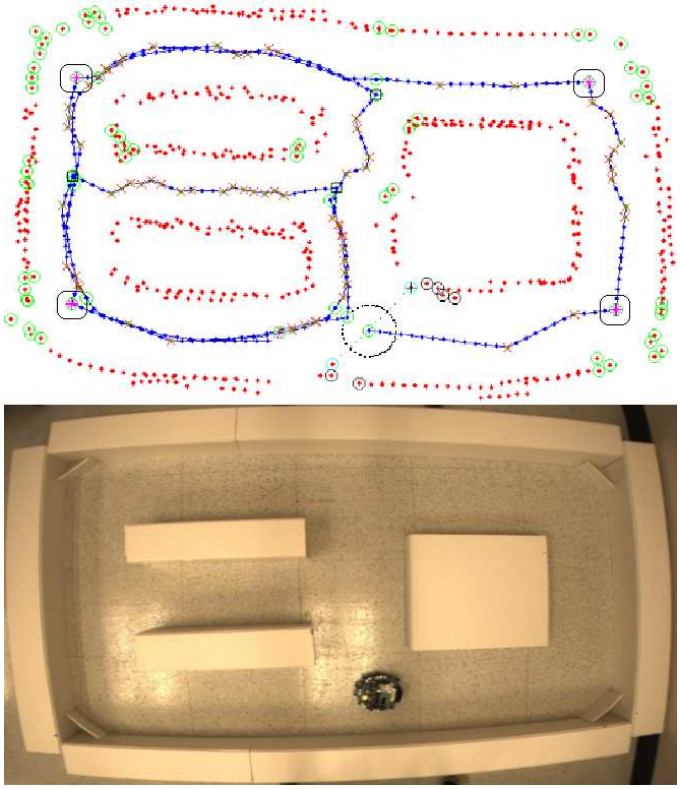

Fig. 12 Three rectangular obstacles are set up in the workspace. A Khepera III robot successfully constructs the Voronoi diagram in this workspace. Real time MATLAB plot is displayed above the snapshot of corresponding obstacle environment. In the MATLAB plot, a rounded rectangle is drawn around an intersection with a blocked sector.

The robot stores coordinates deduced from its odometry and the IR readings for the points detected on the obstacle boundary. These coordinates on the obstacle boundary are referred to as *obstacle points*. To decrease measurement noise in obtaining obstacle points, sensor data are smoothed by convoluting with a Gaussian kernel [22]. The obstacle points derived from IR sensors are shown in red in the MATLAB plot of Fig. 12. Despite using Gaussian smoothing to reduce measurement noise, obstacle points are still scattered.

To allow the robot to detect a blocked sector using IR sensors (sensor range :$\sim$ 0.11m), we set up a small piece of cardboard at each corner of the workspace. In the MATLAB plot of Fig. 12, green circles centered at obstacle points are used to determine whether a sector is open or blocked. When the robot meets an intersection, green circles appear on the obstacle points that are inside a sector. Hence, by observing the distribution of the green circles, the robot can detect a blocked sector at the intersection. In the MATLAB plot of Fig. 12, an intersection with a blocked sector is marked with a magenta star inside a small circle, and a rounded rectangle is drawn around an intersection with a blocked sec-

tor. This figure shows that there are intersections with blocked sectors located at the corners of the workspace. Each blocked sector has two end points located on the boundary of the workspace.

When a closest point on either side of the robot is selected by error from the scattered obstacle points, the robot may unexpectedly move off the Voronoi edge and head toward the obstacle boundary using (11). Once the robot is too close to an obstacle on one side, it may not detect an obstacle on the other side due to short range limitations ($\sim 0.11m$) posted by IR sensors. In this case, instead of using (11), the reactive control [5] is applied for collision avoidance. When the reactive control is applied, the robot's position is marked with "$\times$" in the MATLAB plot (Fig. 12).

In the case where the robot has to move along the enclosing boundary that has been constructed previously, the robot follows the enclosing boundary using a method similar to those in [14, 18, 36]. First, we let a virtual robot move along the enclosing boundary ahead of the real robot. Then, the real robot keeps moving toward the virtual robot to follow the enclosing boundary. Using the virtual robot approach, the real robot builds a smoother trajectory than the enclosing boundary initially built. In addition, the real robot can follow the enclosing boundary with higher speed, since the robot does not have to process sensor data while it moves toward the virtual robot.

8 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 the boundary expansion algorithms so that the Voronoi diagram structure of an unknown compact area can be constructed in finite time. The 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. Furthermore, we provide an analytic upper bound for the total time spent to construct Voronoi diagrams in a regularized workspace. Simulation and experimental results are provided to demonstrate the effectiveness of both the control law and the exploration algorithms.

Acknowledgements We greatly appreciate Sean Maxon for his assistance in setting up the experiments and analyzing data. We thank Justin Shapiro for preparing the Khepera III robots. The research is supported by ONR grants N00014-08-1-1007 and N00014-09-1-1074, and NSF grants ECCS-0845333(CAREER) and CNS-0931576.

References

1. Ahn S, Doh NL, Chung W, Nam S (2008) The robust construction of a generalized Voronoi graph (GVG) using partial range data for guide robots. *Industrial Robot: An International Journal* 35:259-265
2. Aurenhammer F (1991) Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys* 23:345-405
3. Bandyopadhyay S, Coyle EJ (2003) Minimizing communication costs in hierarchically clustered networks of wireless sensors. *Wireless Communications and Networking* 2:1274-1279
4. Bhattacharya P, Gavrilova ML (2008) Roadmap-based path planning - using the Voronoi diagram for a clearance-based shortest path. *IEEE Robotics and Automation Magazine* 15:58-66
5. Brooks RA (1999) Cambrian Intelligence: The Early History of the New AI, MIT Press
6. Brown G (1965) Point density in stems per acre. *Newzealand Forestry Service Research Notes* 38:1-11
7. Choset H, Burdick J (2000) Sensor-based exploration: the hierarchical generalized Voronoi diagram. *The International Journal of Robotics Research* 19:96-125
8. Choset H, Konukseven I, Burdick J (1996) 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, pp. 241-248
9. Choset H, Konukseven I, Rizzi A (1997) Sensor based planning: a control law for generating the generalized Voronoi graph. In *Proc. of 8th International Conference on Advanced Robotics*, CA, USA, pp. 333-338
10. Choset H, Nagatani K (2001) Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation* 17:125-137
11. Choset H, Walker S, Eiamsa-Ard K, Burdick J (2000) Incremental construction of the hierarchical generalized Voronoi graph. *The International Journal of Robotics Research* 19:126-148
12. Cortés J, Martínez S, Karatas T, Bullo F (2004) Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation* 20(2):243-255
13. Du Q, Faber V, Gunzburger M (1999) Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review* 41:637-676
14. Egerstedt M, Hu X, Stotsky A (2001) Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control* 46:1777-1782
15. Fortune SJ (1987) A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2:153-174
16. Garrido S, Moreno L, Blanco D, Martin F (2007) Exploratory navigation based on Voronoi transform and fast marching. In *IEEE International Symposium on Intelligent Signal Processing*, Xiamen, China, pp. 1-6
17. Khalil HK (2002) Nonlinear Systems (3rd ed), Prentice Hall
18. Kim J, Zhang F, Egerstedt M (2009) Curve tracking control for autonomous vehicles with rigidly mounted range sensors. *Journal of Intelligent and Robotic Systems* 56:177-198
19. Klein R (1988) Abstract Voronoi diagrams and their applications. *Computational Geometry and its Applications* 333:148-157
20. Klein R (1990) Concrete and Abstract Voronoi diagrams, Springer
21. Lavalle SM (2006) Planning Algorithms, Cambridge University Press

22. Lindeberg T (1990) Scale-space for discrete signals. *IEEE Transactions of Pattern Analysis and Machine Intelligence* 12(3):234-254
23. Martínez S, Cortés J, Bullo F (2007) Motion coordination with distributed information. *IEEE Control Systems Magazine* 27(4):75-88
24. Mead R (1966) A relation between the individual plant-spacing and yield. *Ann. of Bot., N. S.* 30:301-309
25. Mondada F, Franzi E, Ienne P (1993) Mobile robot miniaturisation: a tool for investigation in control algorithms. In *Proc. of the Third International Symposium on Experimental Robotics*, Kyoto, Japan, pp. 501-513
26. Nagatani K, Choset H (1999) 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, pp. 1687-1692
27. Newman D (1982) The hexagon theorem. *IEEE Transactions on Information Theory* 28:137-139
28. Okabe A, Boots B, Sugihara K (1992) Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Wiley
29. Rao NSV, Stoltzfus N, Iyengar SS (1991) A retraction method for learned navigation in unknown terrains for a circular robot. *IEEE Transactions on Robotics and Automation* 7:699-707
30. Reddy PV, Justh EW, Krishnaprasad PS (2006) Motion camouflage in three dimensions. In *Proc. of 45th IEEE Conf. on Decision and Control*, San Diego, CA, USA, pp. 3327-3332
31. Samson C (1995) Control of chained systems: Application to path-following and time-varying point-stabilization of mobile robots. *IEEE Transactions on Automatic Control* 40:64-77
32. Sudha N, Nandi S, Sridharan K (1999) A parallel algorithm to construct Voronoi diagram and its VLSI architecture. In *Proc. of IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, pp. 1683-1688
33. Svec P (2007) Using methods of computational geometry in robotics. PhD Thesis, Brno University of Technology, Brno, Czech Republic
34. Wein R, Berg JP, Halperin D (2005) The visibility–Voronoi complex and its applications. In *Proc. of the Twenty-first Annual Symposium on Computational Geometry*, Pisa, Italy, pp. 63-72
35. Zhang F, Fratantoni DM, Paley D, Lund J, Leonard NE (2007) Control of coordinated patterns for ocean sampling. *International Journal of Control* 80:1186-1199
36. Zhang F, Justh E, Krishnaprasad PS (2004) Boundary following using gyroscopic control. In *Proc. of 43rd IEEE Conf. on Decision and Control*, Atlantis, Paradise Island, Bahamas, pp. 5204-5209
37. Zhang F, Leonard NE (2007) Coordinated patterns of unit speed particles on a closed curve. *Systems and Control Letters* 56:397-407
38. Zhang F, O'Connor A, Luebke D, Krishnaprasad PS (2004) Experimental study of curvature-based control laws for obstacle avoidance. In *Proc. of IEEE International Conf. on Robotics and Automation*, New Orleans, LA, USA, pp. 3849-3854

Biography

June 5, 2010

Jonghoek Kim is currently a graduate research assistant and Ph.D. candidate at the School of Electrical and Computer Engineering in Georgia Institute of Technology. His research focuses on developing motion control law and motion planning algorithms for robotic sensor networks and multi-agent system. Jonghoek Kim received his M.S. in Electrical and Computer Engineering from Georgia Tech in 2008 and his B.S. in Electrical and Computer Engineering from Yonsei university, South Korea in 2006.

Fumin Zhang is an Assistant Professor in the School of ECE of Georgia Institute of Technology since 2006. He worked as a lecturer and postdoctoral research associate in the Mechanical and Aerospace Engineering Department of Princeton University from 2004 to 2006. He obtained the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Maryland in 2004, College Park, where he also worked for the Institute for Systems Research. His B.S. and M.S. degrees are from Tsinghua University in Beijing in 1995 and 1998 respectively. Dr. Zhang founded the research and teaching program in the fields of robotics and control at Georgia Tech Savannah Campus. His major research focus includes design and control of marine robots and mobile sensor networks, battery modeling and control, and theoretical foundations for cyber-physical systems. He received the CAREER award from NSF in 2009, and the YIP award from ONR in 2010.

Magnus Egerstedt is an Associate Professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology, where he has been on the faculty since 2001. Egerstedt received the M.S. degree in Engineering Physics and the Ph.D. degree in Applied Mathematics from the Royal Institute of Technology in 1996 and 2000 respectively, and he spent 2000-2001 as a Postdoctoral Fellow at Harvard University. Dr. Egerstedt's research interests include hybrid and optimal control, with emphasis on motion planning, control, and coordination of mobile robots, and he has authored over 200 papers in these areas. He serves as Editor for Electronic Publications for the IEEE Control Systems Society and as Associate Editor for the IEEE Transactions on Automatic Control. Magnus Egerstedt is a Senior Member of the IEEE, he received the ECE Junior Faculty Member Award in 2005, and the CAREER award from the National Science Foundation in 2003.

***Author Photographs**

Click here to download high resolution image

***Author Photographs**

Click here to download high resolution image

***Author Photographs**

Click here to download high resolution image