Theoretical Computer Science 1010 (2024) 114714

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Boundary sketching with asymptotically optimal distance and rotation ☆

Varsha Dani [a], Abir Islam [b,*], Jared Saia [b]

[a] Department of Computer Science, Rochester Institute of Technology, Rochester, NY, United States
[b] Department of Computer Science, University of New Mexico, Albuquerque, NM, United States

ARTICLE INFO

Keywords:
Boundary sketch
Robotics
Drones
Distributed algorithms
Euclidean plane

ABSTRACT

We address the problem of designing a distributed algorithm for two robots that sketches the boundary of an unknown shape. Critically, we assume a certain amount of delay in how quickly our robots can react to external feedback. In particular, when a robot moves, it commits to move along path of length at least λ, *or* turn an amount of radians at least λ for some positive $\lambda \leq 1/2^6$, that is normalized based on a unit diameter shape. Then, our algorithm outputs a polygon that is an ϵ-sketch, for $\epsilon = 8\sqrt{\lambda}$, in the sense that every point on the shape boundary is within distance ϵ of the output polygon. Moreover, our costs are asymptotically optimal in two key criteria for the robots: total distance traveled and total amount of rotation.
Additionally, we implement our algorithm, and illustrate its output on some specific shapes.

1. Introduction

What if a robot cannot react instantaneously? In particular, suppose a robot alternates between (1) analyzing past sensor data in order to plan motion of some minimum amount; and (2) executing that plan and gathering new data. Thus, some small, but finite time elapses between first sensing data; and then planning motion.

Now imagine we want such robots to traverse the boundary of an unknown shape in the Euclidean plane. The robots know nothing about the shape in advance, and can only gather local information as they traverse the shape boundary. If the boundary is a continuous curve, efficiently tracing the exact boundary seems challenging. Instead, our goal is to obtain an ϵ-*sketch*: a traversal curve with the property that every point in the actual boundary is within distance ϵ of some point of the sketch; ϵ will be related to the parameter giving the minimum amount a robot can move.

Finally, we want to obtain this ϵ-sketch "efficiently". Unfortunately, for most efficiency measures like time or energy usage, cost is a complicated function of the path traveled, since it must account for both angular and linear momentum. This makes it hard to devise an algorithm with provable asymptotic bounds. Instead, prior work generally either provably optimizes at most one parameter related to efficiency, such as amount turned [23], or distance traveled [31].

In this paper, we take a different approach. Our goal is a bicriteria: minimize both (1) distance traveled, and (2) amount turned. Rather serendipitously, we show that using 2 robots it is possible to asymptotically minimize both criteria. This has broad implications

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

* Corresponding author.
E-mail addresses: varsha.dani@rit.edu (V. Dani), abir@cs.unm.edu (A. Islam), saia@cs.unm.edu (J. Saia).

https://doi.org/10.1016/j.tcs.2024.114714
Received 17 July 2023; Received in revised form 10 May 2024; Accepted 21 June 2024

Available online 28 June 2024

for minimizing a large class of efficiency measures. In particular, our algorithm is also asymptotically optimal for any efficiency function that is polynomial in distance traveled and/or amount turned.

Novelty of Result. The novelty of our results is thus three-fold. First, we handle non-zero robot reaction time and also non-instantaneous sensor measurements. Thus, we improve over control-theoretic results which assume instantaneous reaction time, and instantaneous and continual measurements of quantities such as boundary gradient [16,17,20,34], boundary distance [15,21,32,33], or field measurements [1,6,7]. Second, we assume no a priori shape knowledge. Thus we improve over "robotic coverage" results [8,23,31], which assume a priori knowledge of the boundary. Finally, we asymptotically optimize two key criteria: distance traveled and amount turned. Thus, we improve over results [23,31] that provably only optimize only one such criterion.

1.1. Problem statement

We consider the problem of approximately traversing the boundary of an unknown shape in the Euclidean plane, using two robots.

Problem parameters The diameter of the shape is normalized so that it is 1 unit. Our model depends critically on a parameter $\lambda < 1/2^6$, which describes both the "smoothness" of the shape boundary and the reaction time of the robots as described below.

The robots We make the following assumptions about the two robots.

- Every time a robot moves, it must commit to traveling a path that has distance of at least λ, *or* turning at least λ radians.
- At any point in time, each robot knows its location and whether it is inside or outside the shape. The robots are both initially located a distance of at most $\sqrt{\lambda}$ from the shape boundary.
- When a robot crosses the shape boundary, it learns the gradient at the crossing point.[1]
- The robots can instantaneously communicate with each other.

Our robot model can thus be seen as a variant of the synchronous, unbounded memory case of the Look-Compute-Move or OBLOT model, described in [13]. Two key differences are: (1) our robots "sense" during a cycle instead of "look" at the end of the cycle: they sense when they have traversed the boundary and also sense the gradient at that crossing point; and (2) our robots communicate via shared memory.

The boundary The boundary of the shape is a *curvilinear polygon*,[2] which informally is a closed, non-intersecting loop consisting of a finite number of curves, connected at vertices. Curvilinear polygons include all shapes with boundaries whose gradients are continuous at all but a finite number of points; for example, shapes defined by unions of Gaussians and polygons. They also seem to be the most general shape for which the total rotation of the shape is well-defined.

We make the following additional assumptions about the shape boundary.

- The intersection of the boundary with any ball of a radius $4\sqrt{\lambda}$ centered on a point of the boundary contains *exactly one path component.*[2]
- The vertices of the boundary are at least $\sqrt{\lambda}$ distance apart from each other.
- The boundary is twice continuously differentiable except at the vertices.

Our goal Our goal is to use the robots to estimate the boundary in the form of an ϵ-sketch, while minimizing both distance traveled and the amount turned by the robots.

1.2. Main result

Our main result is given in the following theorem.

Theorem 1. *For any positive $\lambda < 1/2^6$, there exists an algorithm that uses 2 robots to compute an ϵ-sketch of the boundary, for $\epsilon = 8\sqrt{\lambda}$. Moreover the algorithm requires the robots to travel a total distance and rotate a total amount that are both asymptotically optimal.*

As a corollary we can use this ϵ-sketch to estimate the area of the shape.

Corollary 1. *Our algorithm can estimate the area of the shape up to an additive error of $O(\ell\sqrt{\lambda})$, where ℓ is the perimeter of the shape.*

[1] A robot can consider the last gradient encountered in any path of length λ, so estimation of the gradient at the crossing can be computed efficiently (Details in Section 4.1).

[2] These terms are formally defined in Section 3.1 in Definition 5 and Definition 7.

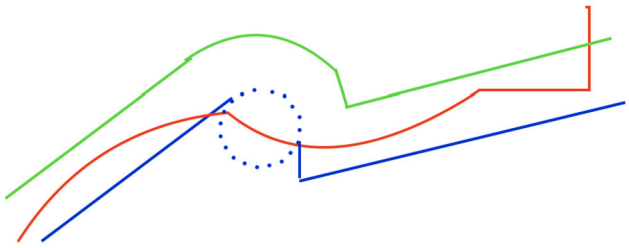

Fig. 1. Illustration of BOUNDARY-SKETCH and CROSS-BOUNDARY. Red curve indicates the shape boundary, blue and green curves indicate the trajectory of the robots that sandwich the boundary. Notice that upon crossing, a dotted blue line (illustrating CROSS-BOUNDARY) indicates a change in direction and step length as discussed in this section. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

1.3. Technical overview

We now give some intuition behind our algorithm and the proof of Theorem 1.

BOUNDARY-SKETCH *intuition* Our algorithm works by trying to ensure a *sandwich invariant*: the robots are traveling in parallel lines on both sides of the boundary. When a robot crosses the boundary, this invariant fails since both robots are now on the same side of the shape. We want the robot that crossed to go back to the other side of the shape in order to reestablish the sandwich invariant. The subroutine CROSS-BOUNDARY performs this function.

The main idea in CROSS-BOUNDARY is to use the boundary gradient learned at the crossing point, to guide the robot back to the other side of the shape and reestablish the sandwich invariant. In CROSS-BOUNDARY, the crossing robot successively takes small steps at a gradually increasing offset from the gradient at the last crossing. The angular offset is in the direction (clockwise or counterclockwise) of the shape boundary. Essentially the robot travels a regular polygon that approximates a small circle, until it crosses the boundary again. After the crossing, the robot reorients its direction so that both robots are moving in parallel lines that sandwich the boundary. See Fig. 1. By repeatedly re-establishing the sandwich invariant whenever it fails, BOUNDARY-SKETCH progressively computes an ϵ-sketch of the boundary.

BOUNDARY-SKETCH *analysis* Our proof of correctness requires tools from real analysis, differential geometry and topology. A main technical challenge is the proof that BOUNDARY-SKETCH produces an ϵ sketch, for $\epsilon = 8\sqrt{\lambda}$. Key milestones in this proof include lemmas showing that the sketch exists; it does not self-intersect; and that the sketch and the shape boundary are "close". We use proof by contradiction extensively to show these results. In particular, we repeatedly construct balls of radius $4\sqrt{\lambda}$ that violate the *path component* assumption unless our desired result holds.

Optimality of distance traversed This part of the asymptotic analysis is relatively straightforward. First, we claim when the sandwich invariant fails, the robots at the end of CROSS-BOUNDARY 1) either cover $\Omega(\sqrt{\lambda})$ distance of the shape boundary or 2) traverse a small distance $O(\sqrt{\lambda})$ between successive instances of Case 1 during a number of executions of CROSS-BOUNDARY. This is proven in Lemma 15, which immediately shows that since the former Case 1 occurs at most $O(\ell/\sqrt{\lambda})$ times, the robots traverse $O(\ell)$ distance to restore the sandwich invariant.

Second, the robots take the shortest path when the sandwich invariant holds, since they move in a straight line parallel to each other, they also traverse $O(\ell)$ distance in this case. The optimality of distance traversed follows by combining these two facts.

Optimality of rotation To prove bounds on rotation, we need to introduce additional formal definitions in Section 3.1, and develop a few helper lemmas in Section 3.2. Our first result is an application of Rolle's Theorem to show the existence, between any points x and y on the shape boundary, of a tangent line somewhere on the boundary between these points that is parallel to the line joining x and y (See Lemma 5). This result has multiple applications including proving two key lemmas, Lemmas 9 and 19. These lemmas were proven via a reduction from the problem for general shapes to shapes that are a polygon. The case of a polygon is one that we can handle easily in the first few lemmas in Sections 3.2 and 3.4.

Lemma 9 is our first key lemma about our unit-diameter shape. It states that the perimeter of our shape is asymptotically bounded by the total "rotation" in the boundary. In particular, it states that $\ell = O(\phi)$, where ℓ is the shape perimeter and ϕ is the boundary rotation, i.e. the total amount a single robot would rotate if it could follow the shape boundary exactly.

The proof of Lemma 9, requires usage of the property of *uniform continuity* of the curvature (a fact that we prove using *continuity* of the curvature along with some topological properties) to split the curve into a finite number of segments, whose endpoints we define to be vertices of a certain polygon. Next, to compare the perimeter of the shape against the perimeter of this polygon, we borrow a key result in differential geometry from [10] stated as Lemma 6. This lemma from differential geometry compares the path length of a curve with bounded curvature against the length of a line segment connecting two endpoints of that curve, and shows that the former is bounded by a constant times the latter. The other case of unbounded curvature is easy to handle from the definition of total rotation in terms of curvature.

Finally, to compare the total rotation of the polygon against the total rotation of the shape boundary, we recall Lemma 5, which says that the shape has at least some point with a boundary gradient that is parallel to the respective side of the polygon. Thus, the total rotation of the polygon is a lower bound on the total rotation of the shape boundary. Thus, we conclude that the shape boundary rotates at least as much as the constructed polygon boundary.

Lemma 19 is another key lemma for bounding the robot rotation. Lemma 19 bounds the number of times the robots make a turn of $\sqrt{\lambda}$ radians during CROSS-BOUNDARY. Once again, we consider the case where the shape boundary between crossings is a polygon first, and then apply Lemma 5 to derive the asymptotics for the general case. Next, we multiply this bound with the rotation angle $\sqrt{\lambda}$ to bound the overall rotation during all executions of CROSS-BOUNDARY.

Lemma 9 handles an intermediate step where total rotation during CROSS-BOUNDARY include the term ℓ and Lemma 19 handles the rest of the analysis of CROSS-BOUNDARY. Together, these two lemmas prove the optimality of rotation by the robots.

1.4. Related work

Application Domains. Robot exploration of a shape is a long-standing problem, which has exploded in popularity recently with the advent of drones and other autonomous devices. Application domains are numerous, running the gamut from surveillance of: forest fires [4,5]; harmful algae blooms [24]; mosquito populations [3,23,30]; oil spills [9,12]; radiation leaks [2]; and volcanic emissions [11].

Boundary Search. Our algorithm assumes that the robots are initially located close to the boundary. The *boundary search problem* instead requires the robots to actually find the boundary. Many algorithms for boundary search have been proposed, techniques used include: random walk [2], spiral search [9], gradient following [27], and finite difference approximation based on partial differential equations [19].

Boundary Following. In *boundary following*, the goal is for the robots to traverse the shape boundary, given that they all initially start close to the boundary. This is the problem addressed in our paper. Many control-theoretic algorithms for boundary following offer provable guarantees that their output converges to the exact boundary under certain assumptions on the boundary shape. However, to the best of our knowledge all such results: (1) assume instantaneous and continuous tracking of some quantity such as boundary gradient or distance to boundary; (2) assume infinitesimally accurate control of the robots; and (3) do not give asymptotic bounds on robot travel time or energy expenditure.

Many such prior results use instantaneous and continuous gradient measurements to control the robots tracking the boundary [16,17,20,34]. Some prior results depend on instantaneous and continuous measurements of other quantities; for example, distance from the boundary [15,21,32,33]; or field measurements defining the shape [1,6,7].

Robotic Coverage. In the *robotic coverage problem*, a robot must visit within some given distance of every point in a target shape. Many variants of this problem are known to be NP-Hard, even with a single robot. Thus, many results either use approximation algorithms or heuristics to optimize some criteria such as distance traveled [31] or amount turned [23]. See [8] for a general overview of results. The problem has been extended to multiple robots [18,29]. Our problem is both easier and harder than the typical robotic coverage problem. It is easier in that we only seek to cover a 1-dimensional boundary, and not a 2-dimensional shape. It is harder in that it is online: no information about the shape is known in advance.

2. Our algorithm: BOUNDARY-SKETCH

Our algorithm BOUNDARY-SKETCH is described in Algorithm 1. In addition to its main subroutine CROSS-BOUNDARY, which is described in Algorithm 3, the algorithms make use of GRADIENTDIRECTION and SYNCHRONIZE which are described in Algorithms 2, 4. We assume an auxiliary function *incomplete* that the robots are capable of, that checks if they have completed a tour around the shape. In addition, by gradient at a given point in this algorithm, we mean a vector with the direction tangent to the shape boundary at the given point. Finally, an initial orientation (clockwise or counter-clockwise) should be selected by the user to determine the direction to start tracing the boundary.

For simplicity of presentation our algorithms are described in a centralized manner, without explicit communication. To parallelize our algorithms, the robots use shared memory. For example, if a robot crosses the boundary during some step, that information is shared with the other robot. (See Figs. 2, 3, 4, 5.)

3. Analysis

In this section, we give the proof of Theorem 1, which assumes that the diameter of the shape is normalized so that it is 1 unit. We divide the analysis into four sections. First, we formalize the notions of curve, path length and total rotations of a curve in our problem model. In addition, we formally state in the language of topology what *path* and *path component* means. The second section establishes some helper lemmas in computational geometry that will be applied in the later sections. Next, we prove that BOUNDARY-SKETCH terminates and outputs an ϵ-sketch of γ, where $\epsilon = 8\sqrt{\lambda}$. Finally, we provide asymptotic analysis of BOUNDARY-SKETCH.

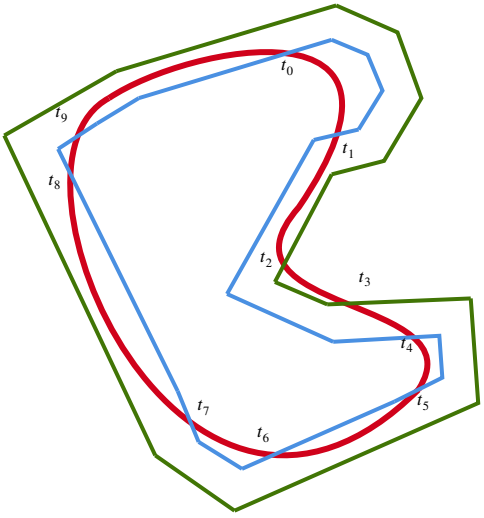

Fig. 2. A high level execution of the BOUNDARY-SKETCH, where the robots cross the boundary over timestamps t_0 through t_9. Here CROSS-BOUNDARY is executed whenever the sandwich invariant fails, in particular around timestamps t_0, t_2, t_4, t_6, t_8. In our formal problem model and analysis, γ denotes the parametrized shape boundary in red and ζ_1, ζ_2 denote the parametrized path of the robots in blue and green.

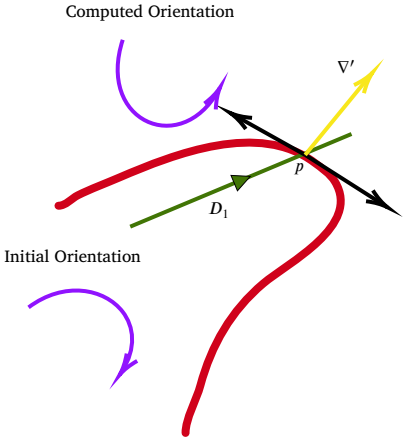

Fig. 3. A sample execution of GRADIENTDIRECTION. After computing the cross product between the direction of D_1 (arrowed green line) and ∇' (arrowed yellow line), the algorithm compares the orientation (counter clockwise) with the initial orientation of the drones (clockwise), and decides to go clockwise from ∇'.

Algorithm 1 Initially, robots are $\sqrt{\lambda}$ apart; one inside and one outside.

procedure BOUNDARY-SKETCH(λ) ▷
  $D_1, D_2 \leftarrow$ the two robots
  $\nabla \leftarrow$ boundary gradient at point of crossing with line segment between D_1 and D_2
  $\alpha \leftarrow \sqrt{\lambda}$
  while Incomplete(D_1, D_2) **do**
    if inside (D_1) XOR inside (D_2) **then**
      D_1 and D_2 both move λ distance in the direction of ∇
    if inside (D_1) = **false** and inside (D_2) = **false then**
      $\alpha \leftarrow \sqrt{\lambda}$
      CROSS-BOUNDARY(D_1, D_2, α)
    elseif inside (D_1) = **true** and inside (D_2) = **true**
      $\alpha \leftarrow -\sqrt{\lambda}$
      CROSS-BOUNDARY (D_2, D_1, α)

Algorithm 2 Finds the direction of the gradient at a given point p of intersection.

```
procedure GRADIENTDIRECTION(p, D_1, D_2)
    Orient ← inside (D_1) XOR inside (D_2)
    Initial Orientation ← false
    ∇' ← Estimated normal vector to the gradient at p
    Sign ← CrossProductSign (Direction vector of D_1, ∇' ).
    if Orient = Initial Orientation then
        Orient ← inside (D_1)
    if Orient = Sign then
        return ∇' + π/2
    else
        return ∇' − π/2
```

Algorithm 3 Reestablishes "Sandwich" Invariant.

```
procedure CROSS-BOUNDARY(D_1, D_2, α)
    p ← last position of D_1 before crossing
    R ← the vertices of the regular polygon including D_1's position with exterior angle √λ and the edge beginning at D_1's position facing the direction of ∇ + α.
    P ← the vertices of the convex hull of R ∪ {p}. For all i : 0 ≤ i ≤ |P| − 1, let P_i be the i-th vertex in this convex hull, ordered such that P_0 = p and P_1 = D_1's
    current position.
    ∇ ← gradient at the last boundary crossing of D_1
    i ← 1.
    while neither robot has crossed the boundary AND i + 1 < |P| do
        D_1 moves to P_{i+1}.
        D_2 moves to closest point from it that is √λ distance away from P_i and orthogonal to ∇ + iα
        i ← i + 1
    while neither robot has crossed the boundary do
        D_1 moves towards point p taking steps of length λ.
        D_2 moves to closest point from it that is √λ distance away from D_1 and orthogonal to D_1's direction.
    if D_2 crossed the boundary then
        SYNCHRONIZE (D_1, D_2)
    else
        ∇ ← the current direction of D_1.
```

Algorithm 4 Ensures the robots are at distance $\sqrt{\lambda}$ from each other and are oriented in the same direction.

```
procedure SYNCHRONIZE(D_1, D_2)
    Path ← the polyline path of D_2 from last crossing of BOUNDARY-SKETCH with the shape till current position.
    ∇ ← the gradient at the last boundary crossing for D_2.
    L_1 ← the line in the direction of ∇ through D_1's position.
    L_2 ← the line in the direction of ∇ through D_2's position.
    if L_1 crosses Path then
        Move D_2 in its current direction until it is √λ distance away from L_1. Change direction to ∇ and take a single step of length λ.
        Move D_1 along L_1 until it is √λ away from D_2.
    else
        Move D_1 in its current direction until it is √λ distance away from L_2. Change direction to ∇ and move until the distance from D_2 is √λ.
        Swap (D_1, D_2).
```

3.1. Formal problem model

The shape is represented by a curve in the Euclidean space. We make use of several definitions, repeated below, about this curve from [25].

Definition 1. A point $\gamma(t)$ of a parameterized curve γ is called a regular point if $\gamma'(t) \neq 0$; otherwise $\gamma(t)$ is a singular point of γ. A curve is regular if all of its points are regular.

Definition 2. A curve $\gamma : [a, b] \to \mathbb{R}^2$ is called a unit-speed curve if for all $t \in [a, b]$, $|\gamma'(t)| = 1$.

The next claim which is Proposition 1.3.6 from [25] relates unit-speed parametrization of curves with regular curves.

Lemma 1. *A parametrized curve has a unit-speed reparametrization if and only if it is regular.*

In what follows, we assume γ is regular unless otherwise stated.

Definition 3. The length of a curve $\gamma : [a, b] \to \mathbb{R}^2$ is defined as,

$$\ell(\gamma) = \int_a^b |\gamma'(t)| dt$$

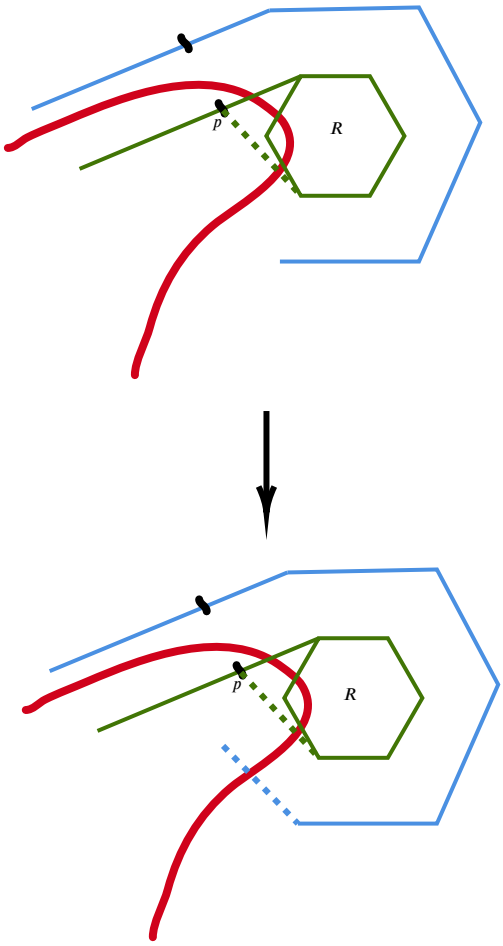

Fig. 4. A high level execution of the CROSS-BOUNDARY subroutine, where the top part indicates the execution of the Algorithm from lines 1 through 10 and the bottom part the rest. Here a small regular polygon R is drawn and then a convex hull P is considered from the vertices of R and the starting position p of the robot moving across the green path. Upon crossing the boundary again, the subroutine SYNCHRONIZE is invoked (see Fig. 5).

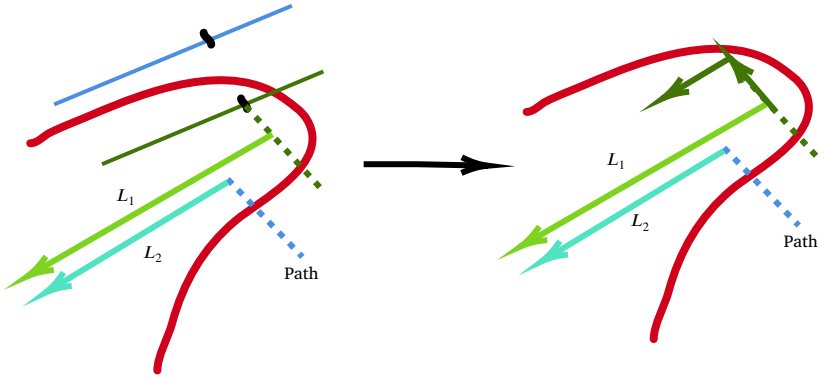

Fig. 5. A sample execution of SYNCHRONIZE. Here dotted blue line indicates a segment of the Path of D_2, with L_1, L_2 as indicated in the algorithm.

Definition 4. If γ is a unit-speed curve with parameter t, its curvature $\kappa(t)$ at the point $\gamma(t)$ is defined to be $|\gamma''(t)|$.

Next we generalize the notion of curve by allowing the possibility of *corners*. More precisely, we use the definition 13.2.1 from [25].

Definition 5. A curvilinear polygon in $\mathbb{R}^2$ is a continuous map $\gamma : \mathbb{R} \to \mathbb{R}^2$ such that, for some real number T and some values $0 = t_0 < t_1 < \ldots < t_n = T$:

1. $\gamma(t) = \gamma(t')$ if and only if $t' - t$ is an integer multiple of T.
2. γ is smooth on each of the open intervals $(t_0, t_1), (t_1, t_2), ..., (t_{n-1}, t_n)$.
3. The one-sided derivatives,

$$\gamma'^{-}(t_i) = \lim_{t \to t_i^-} \frac{\gamma(t) - \gamma(t_i)}{t - t_i}, \gamma'^{+}(t_i) = \lim_{t \to t_i^+} \frac{\gamma(t) - \gamma(t_i)}{t - t_i}$$

exist for all $i = 1, ..., n$ and are non-zero and not parallel.

The points $\gamma(t_i)$ are called the vertices of the curvilinear polygon γ, and the segments of it corresponding to the open intervals (t_{i-1}, t_i) are called its edges. Here T is called the period of γ and if the curve has unit-speed i.e. $|\gamma'(t)| = 1$ for all $t \in \mathbb{R}$, then the length of γ, denoted $\ell(\gamma)$ is T, which is the sum of the length of its edges.

Definition 6. Given a curvilinear polygon γ with vertices at $t_0, t_1, ..., t_n \in [0, T]$ where T is its period, let $\theta_i^{\pm}$ be the angles between $\gamma'^{\pm}(t_i)$ and X-axis. Define $\delta_i = \theta_i^+ - \theta_i^-$ to be the external angle at the vertex $\gamma(t_i)$. The total rotation of γ over the entire period T, denoted $\phi(\gamma)$, is defined to be,

$$\phi(\gamma) = \sum_{i=1}^{n} \delta_i + \int_0^{\ell(\gamma)} |\kappa(t)| dt$$

where we set the speed of γ to be the unit speed.

We will use the following notational practice for simplicity, whenever we mention ℓ, γ without specifying the curve in parenthesis, it means $\ell(\gamma), \phi(\gamma)$ where γ is the shape boundary, otherwise it refers to the curve in parenthesis.

Next, we state a couple of definitions from the Topology textbook of [22].

Definition 7. Given points x and y of a topological space X, a *path* in X from x to y is a continuous map $f : [a, b] \to X$ of some closed interval in the real line into X, such that $f(a) = x$ and $f(b) = y$. Furthermore, $x, y \in X$ are said to be *path connected* if there is a *path* from x to y. In addition, define an equivalence relation between pairs $x, y \in X$ if there is a path in X from x to y. The equivalence classes are called the **path components** of X.

Finally, we define ϵ-sketch.

Definition 8. For $\epsilon > 0$ and a regular curvilinear polygon γ, we say a non self-intersecting polygon P is an ϵ-sketch of γ if every point on γ lies at most an ϵ distance away from P.

Next we begin the analysis with some helper lemmas.

3.2. Helper lemmas

Lemma 2. *ℓ is $\Omega(1)$*

Proof. Since the diameter of the shape is scaled to be 1, $\ell \geq 1$ and the lemma follows. □

Lemma 3. *The number of vertices in γ is at most $\ell / \sqrt{\lambda}$.*

Proof. This is immediate from the assumption that each of the vertices are at least $\sqrt{\lambda}$ distance apart. □

Lemma 4. *If a curve γ is a polygon and ϕ is the total rotation of this polygon, then $\ell(\gamma) \leq \phi$.*

Proof. Suppose the vertices of the polygon P are given by a list of n points $v_1, v_2, ..., v_n$. Then $\ell = \sum_{i=1}^{n} |v_i - v_{i+1}|$ where we set $v_{n+1} = v_1, v_{n+2} = v_2$.

Now fix three successive vertices, v_i, v_{i+1}, v_{i+2} for $1 \leq i \leq n$ and denote them A, B, C respectively. In addition, let $a = |AB|, b = |BC|, c = |CA|, \alpha = \angle BAC, \beta = \angle ABC, \omega = \angle BCA$.

By the law of sines,

$$\frac{a}{\sin \alpha} = \frac{b}{\sin \beta} = \frac{c}{\sin \omega} = 2\rho$$

where ρ is the radius of the circumcircle of the triangle ABC.

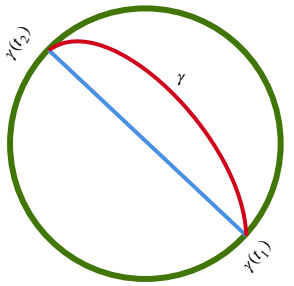

Fig. 6. Figure illustrating Lemma 6.

Since the polygon is bounded by a unit square, $\rho \leq 1$. By the inequality $\sin x \leq x$ for all $x \in \mathbb{R}$, we have,

$$a \leq 2\alpha$$

$$b \leq 2\beta$$

Hence,

$$a + b \leq 2(\alpha + \beta) = 2(\pi - \omega) = 2\phi_i$$

where ϕ_i is the i-th exterior angle of P.

Summing over all $i \in [1, n]$ we get,

$$2\ell(\gamma) = \sum_{i=1}^{n} |v_i - v_{i+1}| + |v_{i+1} - v_{i+2}| \leq 2 \sum_{i=1}^{n} \phi_i = 2\phi$$

This implies,

$$\ell(\gamma) \leq \phi \quad \square$$

Lemma 5. *Let $\gamma : [0, L] \to \mathbb{R}^2$ be a regular curve parametrized by its arc length L such that $\gamma(0) \neq \gamma(L)$. Then there exists $c \in (0, L)$ such that $\gamma'(c)$ is parallel to the line segment joining $\gamma(0)$ and $\gamma(L)$.*

Proof. Let $\gamma(t) = (x(t), y(t))$ for all $t \in [0, L]$ where x, y are differentiable single valued real functions defined over $[0, L]$.

Let u be the vector from $\gamma(0)$ to $\gamma(L)$. That is,

$$u = \gamma(0) - \gamma(0) = (x(L) - x(0), y(L) - y(0))$$

Let v be a vector perpendicular to $\gamma(L) - \gamma(0)$. Since, $\langle u, v \rangle = 0$, we can write,

$$v = (y(0) - y(L), -x(0) + x(L))$$

Now consider the function f defined as follows over $[0, L]$,

$$f(t) = \gamma(t) \cdot v = x(t)(y(0) - y(L)) + y(t)(x(L) - x(0))$$

Observe that, $f(0) = f(L)$. Hence by Rolle's theorem there exists $c \in (0, L)$ such that, $f'(c) = 0$. Since, for all $t \in (0, L)$, $f'(t) = \langle \gamma'(t), v \rangle + \langle \gamma(t), v' \rangle = \langle \gamma'(t), v \rangle$, we conclude, $\langle \gamma'(c), v \rangle = 0$.

This means $\gamma'(c)$ is perpendicular to v. Since u is perpendicular to v as well, we conclude that, $\gamma'(c)$ is parallel to u. $\square$

Next is a lemma found in the following simplified form (p. 272) in [10]. (See Fig. 6.)

Lemma 6. *Let γ be any parametrized curve in $\mathbb{R}^2$ and consider $t_1, t_2 \in [0, \ell(\gamma)]$. For every real number $K > 0$, if the curvature of γ at every point within $[t_1, t_2]$ is not greater than K, then the length of γ over $[t_1, t_2]$ is not longer than half the perimeter of a circle with radius $1/K$ where the end points of its diameter are $\gamma(t_1), \gamma(t_2)$.*

Lemma 7. *Let $\gamma : [a, b] \to \mathbb{R}^2$ be a regular curve and $\kappa : [a, b] \to \mathbb{R}$ be the curvature function of γ. If $|\kappa(t)| \leq 1/\pi$ for all $t \in [a, b]$, then*

$$\int_{t=a}^{b} |\gamma'(t)| dt \leq \frac{\pi |\gamma(b) - \gamma(a)|}{2}$$

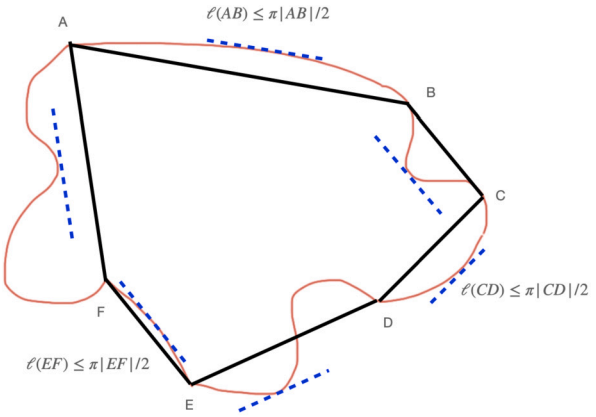

Fig. 7. Figure illustrating the proof for Lemma 9, where $\ell(AB), \ell(CD), \ell(EF)$ indicate the length of the shape (in red color) between the respective endpoints. The black line segments are the sides of the polygon construction and the dotted blue lines indicate the tangents parallel to the respective polygon sides. Here Case 1 is represented by the segments BC, DE, FA while Case 2 is represented by the segments AB, CD, EF.

Proof. Let $A = \gamma(a), B = \gamma(b)$ and $\rho = |\gamma(b) - \gamma(a)|$. Now consider the circle drawn from the midpoint of AB with radius $\rho/2$, where the curvature at every point in this circle is $2/\rho$. Since the shape has unit diameter, $\rho \le 1$. In addition, we have for all $t \in [a, b]$, $|\kappa(t)| \le 1/\pi \le 2/\rho$.

Setting $K = 2/\rho$, we get by Lemma 6,

$$\int_a^b |\gamma'(t)|dt \le \frac{\pi\rho}{2} = \frac{\pi|\gamma(b) - \gamma(a)|}{2}$$

This completes the proof. □

Next we recall a definition from real analysis.

Definition 9. A function $f : X \to Y$ with $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^m$ for $n, m \in \mathbb{N}$ is called uniformly continuous on X if for every real number $\epsilon > 0$, there exists a natural number N such that for every $x, y \in X$,

$$|x - y| < 1/N \implies |f(x) - f(y)| < \epsilon$$

We now state the following lemma that is a simplified form of Theorem 4.19 in [26].

Lemma 8. *Let $f : [a, b] \to \mathbb{R}$ be a continuous mapping with $a, b \in \mathbb{R}$. Then f is uniformly continuous on $[a, b]$.*

Lemma 9. *Let γ be a curvilinear polygon in $\mathbb{R}^2$. Then $\ell(\gamma)$ is $O(\phi)$, where ϕ is the total rotation of γ.*

Proof. Partition into Segments:

Let γ be parametrized by its length, then its period $T = \ell$. Suppose γ has m vertices $\gamma(d_1), \gamma(d_2), ..., \gamma(d_m)$ where $d_i \in [0, \ell]$ for all $i = 1, ..., m$. We also set $d_{m+1} = d_1$.

Since $[d_j, d_{j+1}]$ is closed and κ is continuous over $[d_j, d_{j+1}]$, by Lemma 8, κ is uniformly continuous over $[d_j, d_{j+1}]$. That means for all $x, y \in [d_j, d_{j+1}]$ and $j \in [1, m] \cap \mathbb{N}$, there exists $n \in \mathbb{N}$ such that,

$$|x - y| < (d_{j+1} - d_j)/n \implies ||\kappa(x)| - |\kappa(y)|| \le |\kappa(x) - \kappa(y)| < \frac{1}{2\pi} \tag{1}$$

We now partition each $[d_j, d_{j+1}]$ into at most n segments of the form $[a_k, b_k]$ where $a_k = (k-1)\delta_j/n, b_k = k\delta_j/n, \delta_j = d_{j+1} - d_j$ for $k \in [1, n] \cap \mathbb{N}$. Observe that, by inequality (1) for each of these segments $[a_k, b_k]$, either for all $t \in [a_k, b_k]$, $|\kappa(t)| \ge \frac{1}{2\pi}$ or for all $t \in [a_k, b_k]$, $|\kappa(t)| \le 1/\pi$. We denote these cases by cases 1, 2 in their respective order. (See Fig. 7.)

Finally, over the entire domain $[0, T]$ there are mn segments. Let these segments be indexed by i and let ℓ_i and ϕ_i indicate the perimeter length and the angle turned by the shape in the i-th segment $[a_i, b_i]$.

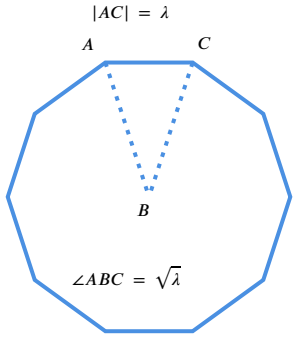

Fig. 8. The regular polygon R constructed in Algorithm 3 with side length λ and exterior angle $\sqrt{\lambda}$.

Case 1:
Since for all $t \in [a_i, b_i]$, $|\kappa(t)| \geq \frac{1}{2\pi}$ then,

$$\phi_i = \int_{a_i}^{b_i} |\kappa(t)| dt \geq \int_{a_i}^{b_i} \frac{1}{2\pi} dt$$
$$\implies \phi_i \geq \frac{1}{2\pi} \ell_i$$
$$\implies \ell_i \leq 2\pi\phi_i$$

Case 2:
Next we handle the other case where the segment $[a_i, b_i]$ has the property that for all $t \in [a_i, b_i]$, $|\kappa(t)| \leq 1/\pi$.

Let P be a polygon consisting of vertices equal to the endpoints of each segment of the shape. For a fixed side of this polygon the endpoints are $\gamma(a_i), \gamma(b_i)$. Let ℓ_P be the perimeter length of P.

By Lemma 7, we have $\ell_i \leq \pi|\gamma(b_i) - \gamma(a_i)|/2$. Hence the total length of the shape over all segments covered by these two cases is at most $\pi\ell_P/2$.

By Lemma 5, there exists a value $c \in [a_i, b_i]$ such that $\gamma'(c)$ is parallel to $\gamma(b_i) - \gamma(a_i)$.

Clearly then $\phi \geq \eta$ where η is the sum of the exterior angles of P.

By Lemma 4, $\ell_P \leq \eta$. This means the length of the perimeter of the shape over all the segments covered by this case is at most $\pi\phi/2$.

Conclusion:
Combining both cases gives $\ell(\gamma) \leq 2\pi\phi$ i.e. $\ell(\gamma) = O(\phi)$. □

3.3. *Correctness of* BOUNDARY-SKETCH

Let ζ_1, ζ_2 be the parametrized curves for the path of the robots D_1 and D_2 in BOUNDARY-SKETCH. Let $t_i \in [0, \ell(\gamma)]$ such that $\gamma(t_i)$ is i-th point of crossing of either robot with the boundary.

Lemma 10. *The regular polygon constructed in Step 3 of Algorithm 3 has diameter at most* $2\sqrt{\lambda}$.

Proof. During Step 10 or Step 13, the regular polygon (see Fig. 8) has side length λ and exterior angle $\sqrt{\lambda}$. Applying the sine law, this polygon has diameter $\frac{\lambda}{\sin\sqrt{\lambda}}$. By the inequality $\sin(2x) \geq x$ for $x \in [0, \pi/4]$ and since $0 < \sqrt{\lambda} < \pi/4$, we have

$$\frac{\lambda}{\sin\sqrt{\lambda}} \leq 2\sqrt{\lambda} \quad \square$$

Lemma 11. *Suppose Algorithm 3 is invoked after crossing the shape for the i-th time. Then $\gamma(t_{i+1})$ is at most $3\sqrt{\lambda}$ distance away from the nearest robot for all invocations of Algorithm 3. In addition, $|\gamma(t_i) - \gamma(t_{i+1})| \leq 3\sqrt{\lambda}$ and the nearest robot traverses no more than $3\sqrt{\lambda}$ distance during the execution of Algorithm 3.*

Proof. First we show that $\gamma(t_{i+1})$ is at most $3\sqrt{\lambda}$ distance away for all invocations of Algorithm 3. If $\gamma(t_{i+1})$ is on the boundary of the regular polygon, then it is at most $2\sqrt{\lambda}$ distance away by Lemma 10. Otherwise by triangle inequality it is at most, $2\sqrt{\lambda} + \lambda < 3\sqrt{\lambda}$ distance away, where the first term is the distance from last visited vertex to the starting vertex of the polygon and the second term is the distance from the starting vertex to $\gamma(t_i)$, which are bounded by the diameter of the regular polygon and step length respectively.

Furthermore, following the argument above, $|\gamma(t_i) - \gamma(t_{i+1})| \leq 3\sqrt{\lambda}$ and the nearest robot traverses no more than $3\sqrt{\lambda}$. □

Lemma 12. *During the While loop of Algorithm 1,* BOUNDARY-SKETCH *maintains a distance of at most $\sqrt{\lambda}$ between each of the robots and the shape boundary throughout all executions of Step 7.*

Proof. This is immediate from the assumption that the robots maintain a distance of $\sqrt{\lambda}$ between them and that the shape is sandwiched there. □

Lemma 13. BOUNDARY-SKETCH *maintains a distance of at most $8\sqrt{\lambda}$ between the shape boundary and each of the robots.*

Proof. By Lemma 12, throughout all executions of Step 7 inside the While loop in Algorithm 1, every point of γ is at a distance of at most $\sqrt{\lambda}$ from ζ_1 and ζ_2.

Suppose here Algorithm 3 is invoked after crossing the shape for the i-th shape. We will show that over the interval $[t_i, t_{i+1}]$, γ is always at most $7\sqrt{\lambda}$ distance away from the nearest robot.

First by Lemma 11 $\gamma(t_{i+1})$ is at most $3\sqrt{\lambda}$ distance away for all invocations of Algorithm 3. In addition, $|\gamma(t_i) - \gamma(t_{i+1})| \leq 3\sqrt{\lambda}$.

Now define $d(x) = |\gamma(t_{i+1}) - \gamma(x)|$ for all $x \in [t_i, t_{i+1}]$. We claim that $d(x) \leq 4\sqrt{\lambda}$ for all $x \in [t_i, t_{i+1}]$. If not, consider a ball B of radius $4\sqrt{\lambda}$ centered at $\gamma(t_{i+1})$. Observe that the path from $\gamma(t_i)$ to $\gamma(t_{i+1})$ must be contained in B or else we will have two different sections that are disjoint inside this ball. This contradicts our *path component* assumption.

That means we can get to $\gamma(t_{i+1})$ first with at most $3\sqrt{\lambda}$ distance traversal by Lemma 11 and then from $\gamma(t_{i+1})$ to the respective point, which is at most $4\sqrt{\lambda}$ distance away by the above argument. Finally, noting that the robots are apart by at most $\sqrt{\lambda}$ distance and by triangle inequality, the lemma follows. □

Lemma 14. *ζ_1, ζ_2 do not self-intersect.*

Proof. We will prove this for ζ_1, the proof is identical for ζ_2.

Suppose there exists u, v such that $\zeta_1(u) = \zeta_1(v)$ and $u \neq v$.

Observe that, unless crossed γ is always on the same orientation (clockwise or counterclockwise) from D_1 and opposite otherwise.

Without loss of generality, assume that γ was on the clockwise direction of D_1 at u. Note that $\zeta_1(u)$ must be inside the shape or else it implies D_1 selected the wrong orientation i.e. Algorithm 2 computed the wrong direction of the gradient.

Let L be the interval of γ with distance at most $2\sqrt{\lambda}$ from $\zeta_1(u)$ on this direction. By Lemma 12, L is nonempty.

Now consider for v an interval R of γ that is on the counterclockwise direction from D_1 at u and that the distance of every point in R from $\zeta_1(u)$ is at most $2\sqrt{\lambda}$. By Lemma 12 R is nonempty.

Now consider the ball B centered at $\zeta_1(u)$ with radius $2\sqrt{\lambda}$. We now show that the intersection of B with L and R are disjoint. If they are not disjoint, they are *path connected* without crossing themselves, since the latter violates the assumption that γ is a simple i.e. non self-intersecting curve.

If they are *path connected*, BOUNDARY-SKETCH crosses this path since L and R are on different orientations (clockwise and counter-clockwise) of $\zeta_1(u)$. But since D_1 upon crossing the shape, chooses to move with the orientation computed by Algorithm 2, it must be that R is on the counterclockwise orientation of $\zeta_1(v)$, a contradiction.

Therefore L and R must be disjoint. Finally, we pick any point $c \in L$ and consider a ball B_1 of radius $4\sqrt{\lambda}$ centered at c. Observe that, L and R can only be connected inside this ball in the same orientation, otherwise it will imply the shape has a bounding box of side length $O(\sqrt{\lambda})$, which contradicts our assumption that the λ is scaled with respect to the diameter of the shape and is at most $1/2^6$.

If L and R connects inside B_1, consider the robot path going in the other orientation. We can extend L and R in this orientation a distance of at most $8\sqrt{\lambda}$ until we can construct another ball B_2 where L and R do not connect. If this construction is not possible, one of the robots must have crossed the boundary and we can construct this ball B_2 with radius $4\sqrt{\lambda}$ centered at that point of crossing, but this contradicts our assumption on *path component*. □

Lemma 15. *For each execution of Algorithm 4, the robots cover $\Omega(\sqrt{\lambda})$ distance of the shape boundary. In addition, the distance traversed by the robots between successive executions of Algorithm 4 and during executions of* CROSS-BOUNDARY *is $O(\sqrt{\lambda})$.*

Proof. The first claim follows immediately since the robot that crosses the boundary changes from D_1 to D_2 and the robots are $\sqrt{\lambda}$ distance apart from each other.

Next, between successive executions of Algorithm 4, robot D_1 may cross the boundary at the end of CROSS-BOUNDARY. The total number of steps robot D_1 can take over this period cannot be more than $2\pi/\sqrt{\lambda}$ since at each step it turns $\sqrt{\lambda}$ and a total turn over a convex path is at most 2π. Since each step is of length λ, the claim follows. □

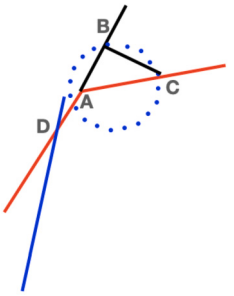

Fig. 9. Figure for Lemmas 16 and 17, the red color indicates the shape boundary, blue color indicates path of the nearest robot, black colored segment is a construction for the proof.

3.4. Asymptotic analysis

Let ϕ_i be the angle the shape turns over $[t_i, t_{i+1}]$ and $f : \mathbb{N} \to \mathbb{N}$ such that $f(i)$ is the number of iterations the While loop inside Algorithm 3 executes in between the robots crossing the shape boundary for the i and $i+1$-th time. We first analyze the case of a polygon.

3.4.1. γ as a polygon

Lemma 16. *Let $j > 1$ be an index such that, after crossing the boundary at point D (Fig. 9), D_1, D_2 are both outside or D_1, D_2 are both inside. Then the number of times the While loop in Step 7 of Algorithm 3 executes before the robots cross the line DB is at most $\frac{\phi_{j-1}}{\sqrt{\lambda}} + 1$.*

Proof. Define ψ_j to be the change of gradient in radians between $\gamma(t_{j-1})$ and $\gamma(t_j)$. Clearly, $\phi_{j-1} \geq \psi_j$.

The vertical distance from the robot at the beginning of the execution of Algorithm 3 to the line DB is at most $\lambda \sin(\psi_j + \sqrt{\lambda})$.

Thus after $\frac{\psi_j}{\sqrt{\lambda}} + 1 \leq \frac{\phi_{j-1}}{\sqrt{\lambda}} + 1$ steps, the robots will cross DB, which concludes the proof. □

Lemma 17. *Suppose γ defines a polygon. Given an instance of crossing the shape at a point D, let β be the first exterior angle of the shape continuing from D. If β is the only exterior angle of γ from D to the next point of crossing and $\sqrt{\lambda} \leq \beta \leq \pi/8$, then the While loop in Step 7 of Algorithm 3 executes at most $8\beta/\sqrt{\lambda}$ times to cover the distance from B to C.*

Proof. Fig. 9 illustrates this lemma where $\angle BAC = \beta$. Note that the radius of the circumcircle of the triangle $\triangle ABC$ is at most the radius ρ of the circumcircle of the regular polygon in the diagram indicated by dotted lines. Given the exterior angle $\sqrt{\lambda}$ and side length λ of the regular polygon, the radius of the circumcircle is $\rho = \frac{\lambda}{2\sin(\sqrt{\lambda})}$. By Lemma 10, $\rho \leq \sqrt{\lambda}$.

Now by the law of sines, $|BC|/\sin\beta = 2\rho$ and this implies $|BC| \leq 2\beta\sqrt{\lambda}$.

Next, the angle β' formed by the chord BC with the center of the regular polygon is at most,

$$2\arcsin\left(\frac{2\beta\sin(\sqrt{\lambda})}{\sqrt{\lambda}}\right) \leq 2\arcsin(2\beta) \leq \frac{4\beta}{\sqrt{1-(2\beta)^2}} \leq \frac{4\beta}{\sqrt{1-\frac{\pi^2}{16}}} \leq 8\beta$$

Next we multiply this angle with the radius to get the arc length between B and C,

$$s = \rho\beta' \leq 8\beta\sqrt{\lambda}$$

Finally, noting that the arc length covered by every step of the robots is at least λ, we get that after $8\beta/\sqrt{\lambda}$ steps from B the robots will cross AC. □

Lemma 18. *If γ is a polygon and if for some $i \leq m$, $\sqrt{\lambda} \leq \phi_i \leq \pi/8$, then $f(i) \leq \frac{8\phi_i}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1$.*

Proof. The trivial case is where Algorithm 3 is not executed at all i.e. $f(i) = 0$.

Observe that the motion of the robots during the execution of Algorithm 3 forms part of the perimeter of a convex polygon with side lengths and exterior angles being λ and $\sqrt{\lambda}$ respectively (except for the first and last sides).

Now suppose there are j vertices of γ defined over $[t_i, t_{i+1}]$. Let these vertices be indexed $\gamma(a_k)$ where $a_k \in [t_i, t_{i+1}]$ for all $k \in [1, j]$. Finally, let β_k be the exterior angle at $\gamma(a_k)$. By Lemma 16, there will be at most $\frac{\phi_{i-1}}{\sqrt{\lambda}} + 1$ before the robot crosses the nearest side of the exterior angle at $\gamma(a_1)$.

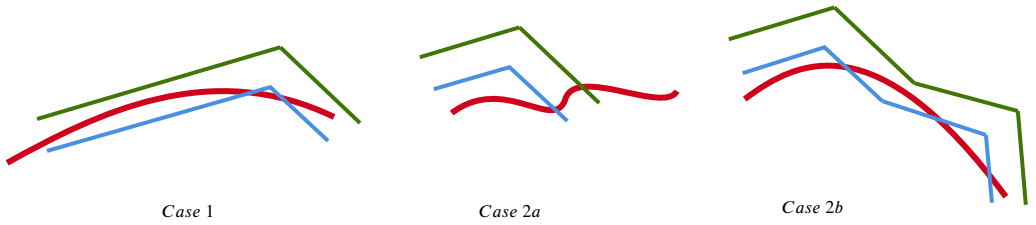

Fig. 10. Illustration of the three cases 1, 2a, 2b in the proof of Lemma 21.

Next, observe that by Lemma 17 the nearest robot to the shape will cross one of the sides of the exterior angle at $\gamma(a_k)$ by at most $8\beta_k/\sqrt{\lambda}$ iterations of the While loop in Algorithm 3.

In addition, the shape boundary turns either in convex or concave manner. If the turn at an index changes from convex to concave or concave to convex, it may actually move the sides of γ closer for the robot and hence the amount the angle β_k contributes to the overall iterations run inside the While loop of Algorithm 3 is at most $8\beta_k/\sqrt{\lambda}$. Therefore,

$$f(i) \leq \sum_{k=1}^{j} 8\frac{\beta_k}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1 = \frac{8\phi_i}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1 \quad \square$$

3.4.2. γ as a curvilinear polygon

Lemma 19. *If $\sqrt{\lambda} \leq \phi_i \leq \pi/8$ for some positive integer $i \leq m$, $f(i) \leq \frac{8\phi_i}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1$.*

Proof. The trivial case is where Algorithm 3 is not executed at all i.e. $f(i) = 0$.

Suppose there are j vertices of γ defined over $[t_i, t_{i+1}]$. Let these vertices be indexed $\gamma(a_k)$ where $a_k \in (t_i, t_{i+1})$ for all $1 \leq k \leq j$. If $j = 0$, select a value $a_1 = (t_{i+1} + t_i)/2$. In addition, let $a_0 = t_i, a_{j+1} = t_{i+1}$.

By Lemma 5, for all $0 \leq k \leq j$, there exists a $c_k \in (a_k, a_{k+1})$ such that $\gamma'(c_k)$ is parallel to the line segment joining $\gamma(a_k)$ and $\gamma(a_{k+1})$. This means if we consider a polygon P with $j+1$ vertices being $\gamma(a_k)$ for $0 \leq k \leq j$, the amount P rotates is at most the amount γ rotates over $[t_i, t_{i+1}]$.

If ϕ_i' is the amount of rotation of P, then by Lemma 18,

$$f(i) \leq \frac{8\phi_i'}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1 \leq \frac{8\phi_i}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1 \quad \square$$

Lemma 20. *If $\pi/8 \geq \phi_i \geq \sqrt{\lambda}$, after resetting ∇ in lines 15 or 17 of Algorithm 3, the robots turn at most $8\phi_i + \phi_{i-1} + \sqrt{\lambda}$ as the algorithm continues to execute Algorithm 1.*

Proof. The angle the robot needs to turn to reorient itself with respect to the boundary just crossed is at most $8\phi_i + \phi_{i-1} + \sqrt{\lambda}$, since the robot orientation itself is no more off than $8\phi_i + \phi_{i-1} + \sqrt{\lambda}$ by Lemma 19. $\square$

Lemma 21. *ζ_1, ζ_2 have finite periods and therefore they intersect the shape finitely many times.*

Proof. Note that during each execution of CROSS-BOUNDARY, since the exterior angle of the polygon R is $\sqrt{\lambda}$, BOUNDARY-SKETCH turns at least $\sqrt{\lambda}$. If $\phi_i > \frac{\pi}{8}$ during execution of Algorithm 3, BOUNDARY-SKETCH turns at most 2π. Thus, the radians turned is bounded by $16\phi_i$. Next, by Lemma 20, the indices i for which $\pi/8 \geq \phi_i \geq \sqrt{\lambda}$, BOUNDARY-SKETCH lower bounds the total radians turned by the shape γ.

Since the total radians turned by γ overall is lower bounded by the amount turned by the robots over these indices, the number of such indices must be finite.

Now consider the indices i such that $\phi_i < \sqrt{\lambda}$. In this case CROSS-BOUNDARY will cross the boundary immediately after taking one step after crossing the boundary on t_i.

There are two cases, the robots sandwiched the boundary prior to crossing on t_i (Case 1) or they did not (Case 2).

Now we analyze the length of γ over $[t_i, t_{i+1}]$ for these two cases.

Case 1: The length of γ over $[t_i, t_{i+1}]$ is lower bounded by λ, since that is the step length of the robots and they move in straight line in parallel while the boundary does not (see Fig. 10).

Case 2: We claim that the length of γ over $[t_i, t_{i+1}]$ is once again lower bounded by λ. To show this, first observe Case 2a where if CROSS-BOUNDARY is called again on t_i then the robots cover $\sqrt{\lambda}$ distance of γ since the boundary has been crossed by both of the robots. The other Case 2b is reduced to a similar situation as before (see Fig. 10), where if we have successive indices $i, i+1$

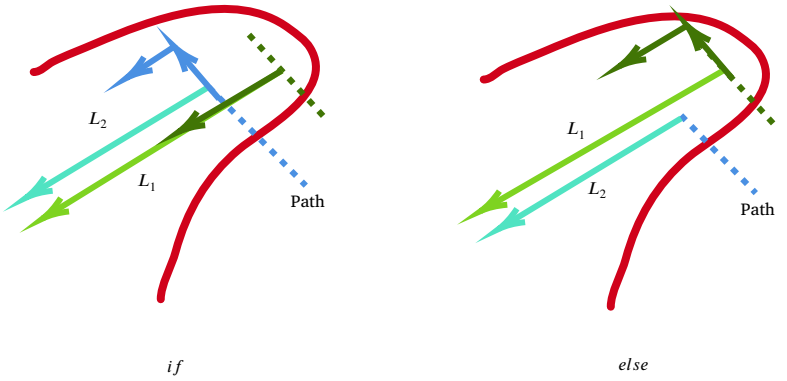

Fig. 11. Illustration of the two cases in SYNCHRONIZE.

for which ϕ_i, ϕ_{i+1} are both less than $\sqrt{\lambda}$, the claim holds. Otherwise we can include the index i among the count of indices with $\phi_{i+1} \geq \sqrt{\lambda}$.

Since the total length of γ is finite, the number of such indices is also going to be finite.

Hence, CROSS-BOUNDARY is executed only finitely many times and BOUNDARY-SKETCH terminates with ζ_1, ζ_2 having a finite period. □

Lemma 22. *Let I be those indices such that, $\sqrt{\lambda} \leq \phi_i \leq \pi/8$ for $i \in I$. Then the total radians turned by the algorithm for the ϕ_i values indexed by I is $O(\phi)$.*

Proof. Observe that, $\sum_{i \in I} \phi_i \leq \phi$. In addition by Lemma 19 $f(i) \leq \frac{8\phi_i}{\sqrt{\lambda}} + \frac{\phi_{i-1}}{\sqrt{\lambda}} + 1$ and by Lemma 20 the angle turned after resetting ∇ in lines 15 or 17 in Algorithm 3 is at most $8\phi_i + \phi_{i-1} + \sqrt{\lambda}$.

Thus the total radians turned by the algorithm for the ϕ_i values indexed by I is at most:

$$\sum_{i \in I} \sqrt{\lambda} f(i) + 8\phi_i + \phi_{i-1} + \sqrt{\lambda} \leq \sum_{i \in I} 16\phi_i + 2\phi_{i-1} + \sqrt{\lambda} = 16\phi + |I|\sqrt{\lambda} = O(\phi)$$

where we note $\phi_i \geq \sqrt{\lambda}$ implies $|I| = O(\phi/\sqrt{\lambda})$. □

Lemma 23. BOUNDARY-SKETCH *terminates with the output curves ζ_1, ζ_2 which are $8\sqrt{\lambda}$-sketches of γ.*

Proof. This follows immediately from Lemmas 13, 14, 21 and Definition 8. □

3.4.3. Lemmas about synchronization

Our final two lemmas show that the other robot do not rotate or traverse asymptotically more than the robot nearest to the boundary. In addition, we discuss briefly the synchronization steps in Algorithm 4.

Observe that in Fig. 11, after reorientating itself to the curve gradient direction, the green and blue curves can become "closer" to each other if they simply turn towards the gradient. This is handled by letting the blue or green curve based robot traverse a little longer, in particular greater than $\sqrt{\lambda} - \lambda > \lambda$ distance and then turn. This synchronization that maintains the distance of $\sqrt{\lambda}$ between the two robots guarantees that between successive executions of Algorithm 4, the robots will cover $\Omega(\sqrt{\lambda})$ distance of the shape boundary, in turn we are able to prove Lemma 15.

Lemma 24. *For each execution of Algorithm 3, D_2 traverses a distance of $O(\sqrt{\lambda})$.*

Proof. In Fig. 12, the line segment AC describes the path of D_1 and D_2 can move either directly from B to D or B to D via E. Now, by Algorithm 3 description, $|AC| = |DE| = \lambda$. Note that, $\angle BAE = \sqrt{\lambda}$ and $|BA| = |AE| = \sqrt{\lambda}$. Thus by the law of sine, $\frac{|EB|}{\sin\sqrt{\lambda}} = \frac{\sqrt{\lambda}}{\sin((\pi - \sqrt{\lambda})/2)}$ Hence,

$$|EB| \leq \frac{\lambda}{\cos(\sqrt{\lambda}/2)} \leq 2\lambda.$$

Thus, the path length of D_2 at each step is at most 3λ.

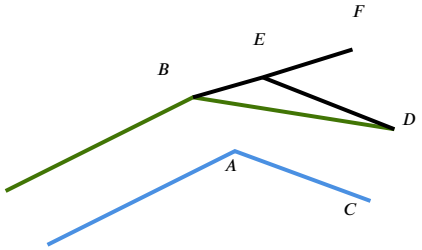

Fig. 12. Figure for Lemmas 24 and 25.

In the last step independent of which robot encountered the shape boundary and whether they are both inside or outside or one inside and one outside, D_2 will traverse at most a constant factor of $\sqrt{\lambda}$ by the above discussion based on Fig. 11. Finally, noting that D_1 traverses $O(\sqrt{\lambda})$ distance according to Lemma 11, D_2 traverses $O(\sqrt{\lambda})$ during each execution of Algorithm 3. □

Lemma 25. *For each execution of Algorithm 3, D_2 does not rotate asymptotically more than D_1.*

Proof. We only need to check that the rotation of the green path towards BE and then ED is asymptotically bounded by D_1's rotation. For brevity, we simply note this follows by elementary euclidean geometric comparison of various angles, beginning from the rotation of the blue path to AC.

Finally, in the last step, both robots turn towards a common direction i.e. gradient at a point of crossing the boundary. □

3.4.4. Proof of the main theorems

Theorem 2. *The robots in* BOUNDARY-SKETCH *traverse a total distance of $O(\ell)$.*

Proof. During Step 7 of Algorithm 1, the robots take the shortest path and thus the distance traversed is bounded by the shape perimeter. Hence overall the total distance traversed during execution of Step 7 is $O(\ell)$.

Now each time Algorithm 4 is executed, the robots cover at least $\Omega(\sqrt{\lambda})$ distance of the shape boundary. This means Algorithm 4 is executed at most $O(\ell/\sqrt{\lambda})$ times. In addition, between successive executions of Algorithm 4 and during executions of CROSS-BOUNDARY the robots traverse a distance at most $O(\sqrt{\lambda})$ by Lemma 15.

Combining we have that $\ell(\zeta_1)$ and $\ell(\zeta_2)$ are both $O(\ell)$. □

Theorem 3. *The total rotation by the robots in* BOUNDARY-SKETCH *is $O(\phi)$.*

Proof. The number of indices $i \leq m$ such that $\phi_i \leq \sqrt{\lambda}$ is $O(\ell/\lambda)$. For each of these indices, the robots rotate $O(\lambda)$ angle in CROSS-BOUNDARY by Lemma 16. Since $\ell = O(\phi)$, for these indices the robots rotate $O(\phi)$.

Now, if $\pi/8 \geq \phi_i \geq \sqrt{\lambda}$, by Lemma 22, the robots rotate a total of $O(\phi)$ radians.

If $\phi_i > \frac{\pi}{8}$ during execution of Algorithm 3, BOUNDARY-SKETCH turns at most 2π. Thus, the radians turned is bounded by $16\phi_i$. In total, across all iterations of Algorithm 3 where $\phi_i > \frac{\pi}{8}$, the robots rotate at most 16ϕ.

In addition, by Lemma 25, the robots asymptotically rotate the same.

Finally, during execution of Step 7 the robots do not rotate at all. Combining all of the above, we have the theorem. □

Corollary 2. *The area estimated by* BOUNDARY-SKETCH *differs from the actual shape area by $O(\ell\sqrt{\lambda})$.*

Proof. BOUNDARY-SKETCH computes the area of the polygon generated by successive positions of one of the robots (say the one that starts from the inside). By Lemma 13, this polygon stays within at most $8\sqrt{\lambda}$ distance from the shape boundary. Hence, the area of the polygon differs from the shape area by at most $O(\ell\sqrt{\lambda})$. □

4. Implementation guide and simulations

4.1. Implementation guide

BOUNDARY-SKETCH is largely agnostic to specific details of its implementation. However, below we make a few suggestions with regards to i) representation of the boundary in practice ii) computation of the gradient iii) possible laws governing the motion of the robots.

4.1.1. Representation of the boundary

Analytically, we have used the formal language of *curvilinear polygon* from differential geometry. In practice, *any* representation can be used that can be mapped to our analytical representation. For example, we can define γ to be any level contour of a surface given by $f : \mathbb{R}^2 \to \mathbb{R}$. By setting a value k, the implicit equation $f(x, y) = k$ then determines γ. In addition, the robots D_1, D_2 can make use of some kind of sensor to measure or estimate the value of f at a given point, which allows them assess if they are inside (e.g. when $f(x, y) > k$) or outside (e.g. when $f(x, y) < k$) of the boundary.

4.1.2. Computation of the gradient

Our algorithm BOUNDARY-SKETCH assumes an oracle access to the gradient. We acknowledge in practice this might be inhibiting. However, we present a suggestion below how one might estimate the gradient using Least Squares (LSQ) Error formulation.

Least squares error formulation Assume we have three points a, b, c that are close together, and that we have the values $f(a)$, $f(b)$, $f(c)$. For the sake of the BOUNDARY-SKETCH, these three points can be said to be successive points of a robot's path with the middle one being where it crosses the shape boundary. Then we estimate the vector (∇' in Algorithm 2) normal to the gradient $\nabla f(b) = (x', y')$ as a vector that minimizes the following sum:

$$(f(a) - f(b) - \nabla f(b) \cdot (a - b))^2 + (f(c) - f(b) - \nabla f(b) \cdot (c - b))^2$$

Setting the rows of a 2×2 matrix A to be $a - b$, $c - b$ respectively and the entries of vector β to be $f(a) - f(b), f(c) - f(b)$ respectively, this problem is formally solving for the vector v such that, $|Av - \beta|^2$ is smallest.

Finally, we utilize the pseudo-inverse method for solving LSQ as discussed in numerous sources e.g. in Section 4.5 of [14].

4.1.3. Motion of the robots

BOUNDARY-SKETCH is additionally agnostic to laws governing the motion of the robots. However, a real-world application domain of particular interest to us is tracing boundaries of volcanic CO2 plumes using drones. In that case, control laws govern the motion of the drones which can be mapped to the various actions necessary in the algorithm, in particular turning a certain angle, moving in tandem and the synchronization steps in Algorithm 4.

Finally, CO2 plumes can be represented by a function $f : \mathbb{R}^2 \to \mathbb{R}$ where $f(x, y)$ is the CO2 measurement at point (x, y) and sensors can be used to estimate $f(x, y)$, thus effectively determining whether a robot is inside or outside.

4.2. Simulations

We present preliminary simulation results (with GitHub source code [28]) to illustrate the precision of ϵ-sketch for various values of λ. These simulations specifically do not perform any physical motion of the robots. However, they do estimate the gradient instead of assuming a perfect oracle access.

To start with, we tested our algorithm for a small number of intersecting gaussians with different variances. Next, we test our algorithm on the boundary of shapes drawn from real world shape data. Both of these simulations demonstrate encouraging convergence of ϵ-sketch to the shape as λ decreases.

4.2.1. Intersecting Gaussians

In our experiments, the shape is an intersection of a few two dimensional gaussians.

Definition 10. In two dimensions, the elliptical gaussian function f for uncorrelated varieties X and Y having a bivariate normal distribution and standard deviations σ_x, σ_y is defined to be,

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left[\frac{(x-\sigma_x)^2}{(2\sigma_x)^2} + \frac{(y-\sigma_y)^2}{(2\sigma_y)^2}\right]}$$

To estimate the gradient at a point of intersection, we used the least squares error (LSQ) problem as suggested above.

Test shape We generate a test shape in MATLAB with four gaussians with centers $(1, 1), (-2, 0), (1, 0), (4, 0)$ and corresponding standard deviations

$$(0.8, 0.8), (1, 1), (1.2, 1.2), (0.7, 0.7).$$

We also ignore the $1/2\pi$ factor in front of the definition.

BOUNDARY-SKETCH *for different* λ For the same shape as above, we run BOUNDARY-SKETCH for λ values of 0.0001, 0.000025, 0.000001. Fig. 13 illustrates the output for $\lambda = 0.0001$ and the other two Figs. 14, 15 demonstrate notable improvement in precision as λ gets smaller.

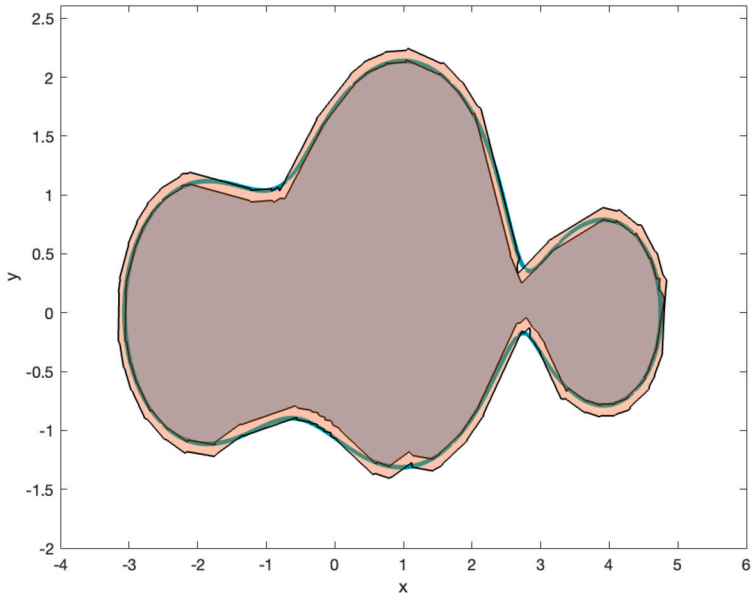

Fig. 13. BOUNDARY-SKETCH for $\lambda = 0.0001$, the green border marks the shape boundary sandwiched by the inner and outer robots.

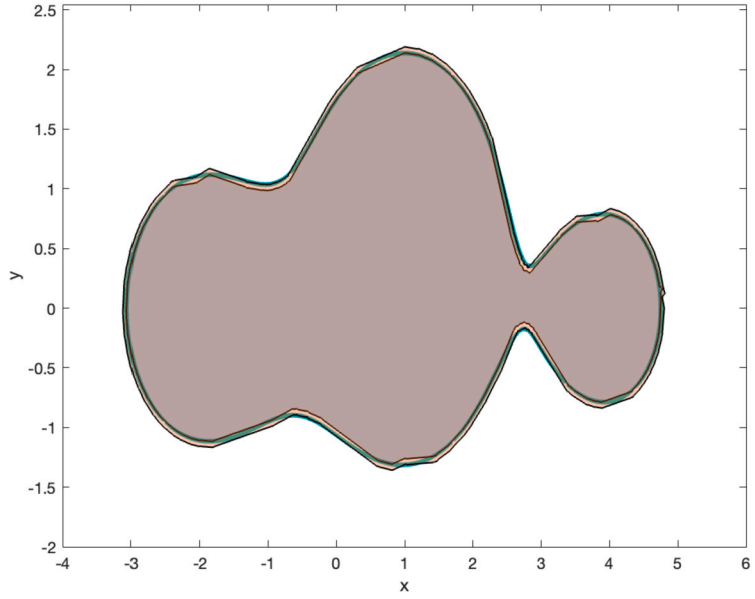

Fig. 14. BOUNDARY-SKETCH for $\lambda = 0.000025$, the green border marks the shape boundary sandwiched by the inner and outer robots.

4.2.2. Real world plume shapes

We also utilize a real world shape from a volcanic plume in La Palma (see Fig. 16) and use a Python program to generate a polytope approximation of it. The gradient of the boundary then is easily found by the gradient of the corresponding side of the polygon. Next we ran BOUNDARY-SKETCH for different values of λ on this shape. Fig. 17 illustrates the output of the algorithm and Figs. 18, 19 demonstrate notable improvement once again in precision as λ gets smaller.

5. Conclusion

We have described a distributed algorithm to enable two robots to traverse the perimeter of a curvilinear polygon. Our algorithm is novel in three key ways. First, it does not assume that the robots can respond instantaneously to sensor data, instead assuming a minimum amount of movement or rotation occurs between motion planning events. Second, it does not assume that the robots have access to instantaneous and continual sensor readings. Finally, our algorithm is simultaneously asymptotically optimal in two criteria: both total distance traveled by the robots, and also total amount turned by the robots.

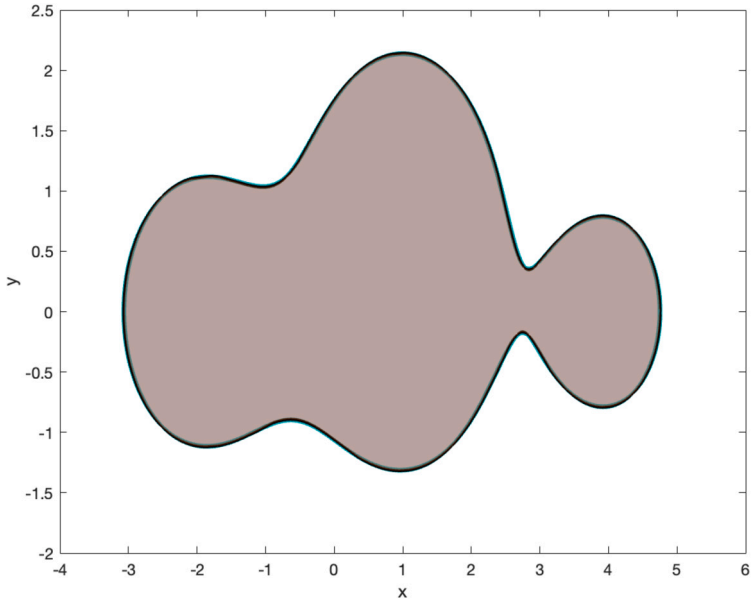

Fig. 15. BOUNDARY-SKETCH for $\lambda = 0.000001$, the green border marks the shape boundary sandwiched by the inner and outer robots.

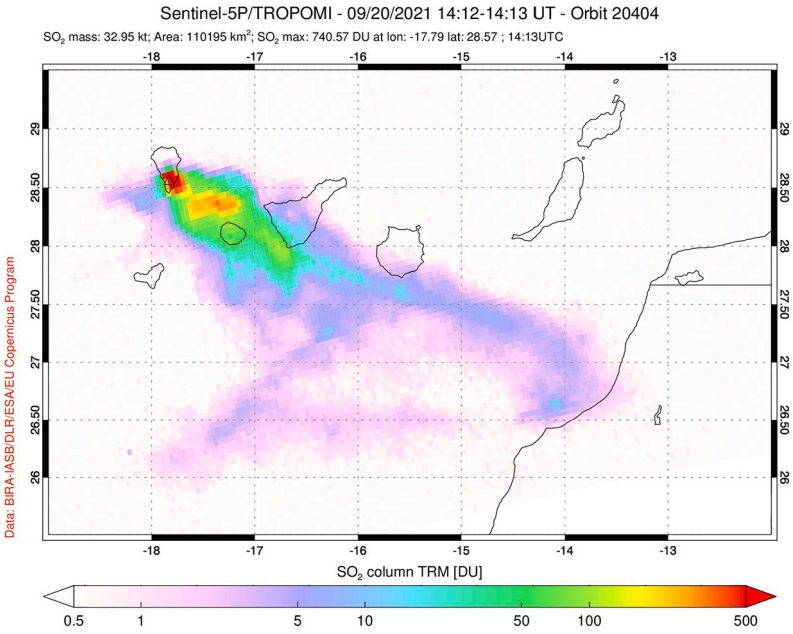

Fig. 16. Volcanic plume in La Palma observed on September 20, 2021.

Several open problems remain including the following. First, lower bounds: even though our algorithm is asymptotically optimal in both distance traveled and distance turned, we still would like to know if 2 robots are strictly necessary. Can a single robot achieve the same asymptotic results? We conjecture the answer is no, even if the single robot is attempting to follow a (unknown) function in the Euclidean plane defined from the values $x = 0$ to $x = 1$. Second, Can we reduce the value of ϵ in the ϵ-sketch returned by our algorithm? In particular, is it possible to obtain a value of ϵ that is $o(\sqrt{\lambda})$? Finally, How does our algorithm perform when deployed in the real world? An application domain of particular interest is tracing boundaries of volcanic CO2 plumes. We are currently adapting BOUNDARY-SKETCH for this domain and expect to publish our results in the near future.

Acknowledgements

We would like to acknowledge the members of the UNM Volcan team for their contributions to this paper.

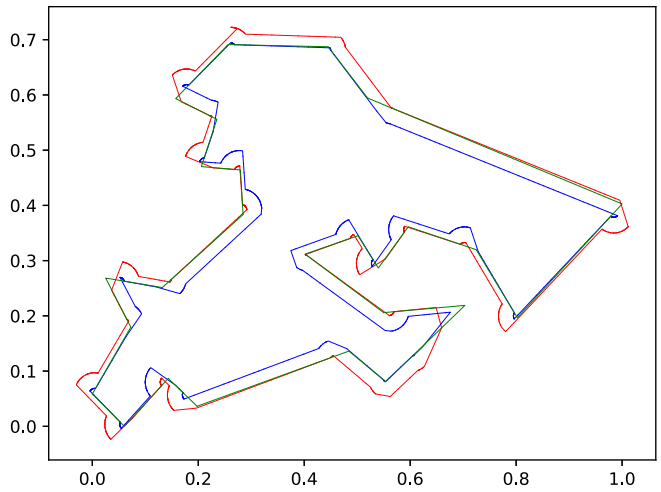

Fig. 17. BOUNDARY-SKETCH for $\lambda = 0.0001$, the green border marks the shape boundary sandwiched by the inner and outer robots (red and blue border).

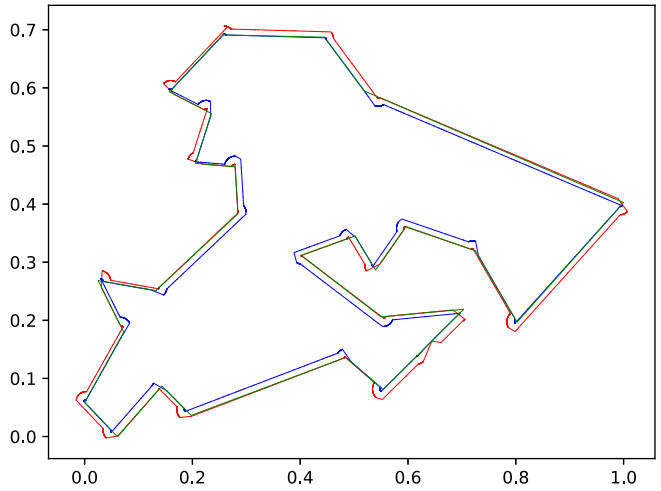

Fig. 18. BOUNDARY-SKETCH for $\lambda = 0.000025$, the green border marks the shape boundary sandwiched by the inner and outer robots (red and blue border).

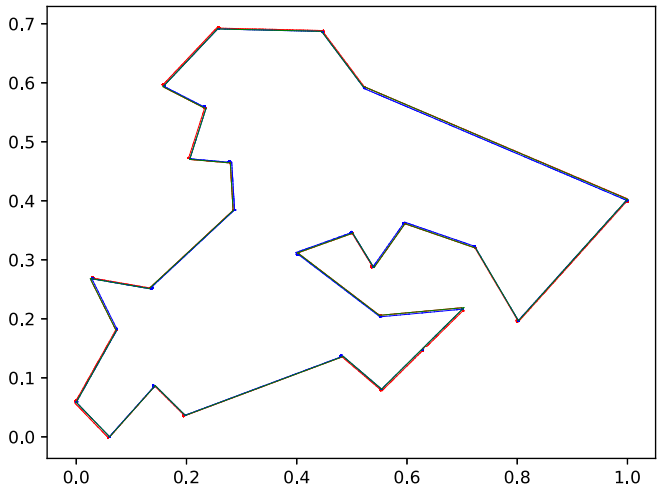

Fig. 19. BOUNDARY-SKETCH for $\lambda = 0.000001$, the green border marks the shape boundary sandwiched by the inner and outer robots (red and blue border).

In particular, John Ericksen contributed code for some of the geometric operations in the Sketch Algorithm and the rendering of the plume, which was used to generate Figures 13, 14, 15, 17, 18, 19. He also contributed ideas through numerous technical discussions that helped inform the simulation of the Sketch Algorithm. We also thank Melanie Moses and Matthew Fricke for numerous discussions and feedback that shaped the development of the paper.

CRediT authorship contribution statement

Varsha Dani: Writing – original draft, Writing – review & editing. **Abir Islam:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Jared Saia:** Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Abir Islam reports financial support was provided by National Science Foundation grant IIS 2024520. Jared Saia reports financial support was provided by National Science Foundation grant IIS 2024520.

References

[1] S. Al-Abri, F. Zhang, A distributed active perception strategy for source seeking and level curve tracking, IEEE Trans. Autom. Control 67 (2021) 2459–2465.
[2] D.J. Bruemmer, D.D. Dudenhoeffer, M.D. McKay, M.O. Anderson, A robotic swarm for spill finding and perimeter formation, Technical Report, Idaho National Engineering and Environmental Lab Idaho Falls, 2002.
[3] G. Carrasco-Escobar, M. Moreno, K. Fornace, M. Herrera-Varela, E. Manrique, J.E. Conn, The use of drones for mosquito surveillance and control, Parasites Vectors 15 (2022) 473.
[4] D.W. Casbeer, R.W. Beard, T.W. McLain, S.M. Li, R.K. Mehra, Forest fire monitoring with multiple small uavs, in: Proceedings of the 2005, American Control Conference, IEEE, 2005, pp. 3530–3535.
[5] D.W. Casbeer, D.B. Kingston, R.W. Beard, T.W. McLain, Cooperative forest fire surveillance using a team of small unmanned air vehicles, Int. J. Syst. Sci. 37 (2006) 351–360.
[6] S. Chatterjee, W. Wu, Cooperative curve tracking in two dimensions without explicit estimation of the field gradient, in: 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, 2017, pp. 0167–0172.
[7] S. Chatterjee, W. Wu, A modular approach to level curve tracking with two nonholonomic mobile robots, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2019, V009T12A051.
[8] H. Choset, Coverage for robotics–a survey of recent results, Ann. Math. Artif. Intell. 31 (2001) 113–126.
[9] J. Clark, R. Fierro, Cooperative hybrid control of robotic sensors for perimeter detection and tracking, in: Proceedings of the 2005, American Control Conference, IEEE, 2005, pp. 3500–3505.
[10] B. Dekster, The length of a curve in space with curvature at most k, American Mathematical Society 79 (2) (1980) 271–278.
[11] J. Ericksen, G.M. Fricke, S. Nowicki, T.P. Fischer, J.C. Hayes, K. Rosenberger, S.R. Wolf, R. Fierro, M.E. Moses, Aerial survey robotics in extreme environments: mapping volcanic co2 emissions with flocking uavs, Frontiers in Control Engineering 3 (2022) 7.
[12] M. Fahad, N. Saul, Y. Guo, B. Bingham, Robotic simulation of dynamic plume tracking by unmanned surface vessels, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 2654–2659.
[13] P. Flocchini, G. Prencipe, N. Santoro, Distributed computing by mobile entities, Current Research in Moving and Computing (2019) 11340.
[14] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[15] M. Hoy, A method of boundary following by a wheeled mobile robot based on sampled range information, J. Intell. Robot. Syst. 72 (2013) 463–482.
[16] M.A. Hsieh, V. Kumar, L. Chaimowicz, Decentralized controllers for shape generation with robotic swarms, Robotica 26 (2008) 691–701.
[17] M. Kemp, A.L. Bertozzi, D. Marthaler, Multi-uuv perimeter surveillance, in: Proceedings of IEEE/OES Autonomous Underwater Vehicles, 2004, pp. 102–107.
[18] S. Koenig, B. Szymanski, Y. Liu, Efficient and inefficient ant coverage methods, Ann. Math. Artif. Intell. 31 (2001) 41–76.
[19] D. Marthaler, A.L. Bertozzi, Collective motion algorithms for determining environmental boundaries, in: SIAM Conference on Applications of Dynamical Systems, 2003, pp. 1–15.
[20] D. Marthaler, A.L. Bertozzi, Tracking environmental level sets with autonomous vehicles, in: Recent developments in cooperative control and optimization, 2004, pp. 317–332.
[21] A.S. Matveev, M.C. Hoy, A.V. Savkin, The problem of boundary following by a unicycle-like robot with rigidly mounted sensors, Robot. Auton. Syst. 61 (2013) 312–327.
[22] J. Munkres, Topology, Prentice-Hall Pearson, 2000.
[23] A. Nguyen, D. Krupke, M. Burbage, S. Bhatnagar, S.P. Fekete, A.T. Becker, Using a uav for destructive surveys of mosquito population, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 7812–7819.
[24] L.H. Pettersson, D. Pozdnyakov, Monitoring of Harmful Algal Blooms, Springer Science & Business Media, 2012.
[25] A. Pressley, Elementary Differential Geometry, Springer, 2012.
[26] W. Rudin, Principles of Mathematical Analysis, McGraw-Hill Education, 1976.
[27] D. Saldana, R. Assunção, M.F. Campos, A distributed multi-robot approach for the detection and tracking of multiple dynamic anomalies, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 1262–1267.
[28] Sketch, Simulation of sketch algorithm, https://github.com/encyclo1/sketchalgorithm, 2023.
[29] D. Spears, W. Kerr, W. Spears, Physics-based robot swarms for coverage problems, Int. J. Intell. Control Syst. 11 (2006) 11–23.
[30] M.C. Stanton, P. Kalonde, K. Zembere, R. Hoek Spaans, C.M. Jones, The application of drones for mosquito larval habitat identification in rural environments: a practical approach for malaria control?, Malar. J. 20 (2021) 1–17.
[31] P. Tokekar, E. Branson, J. Vander Hook, V. Isler, Tracking aquatic invaders: autonomous robots for monitoring invasive fish, IEEE Robot. Autom. Mag. 20 (2013) 33–41.
[32] F. Zhang, S. Haq, Boundary following by robot formations without gps, in: 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 152–157.
[33] F. Zhang, E.W. Justh, P.S. Krishnaprasad, Boundary following using gyroscopic control, in: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), IEEE, 2004, pp. 5204–5209.
[34] F. Zhang, N.E. Leonard, Cooperative filters and control for cooperative exploration, IEEE Trans. Autom. Control 55 (2010) 650–663.