EFFICIENT EVADER DETECTION IN MOBILE SENSOR NETWORKS

HENRY ADAMS\textsuperscript{1}, DEEPJYOTI GHOSH\textsuperscript{2}, CLARK MASK\textsuperscript{3}, WILLIAM OTT\textsuperscript{*}\textsuperscript{,3}, AND KYLE WILLIAMS\textsuperscript{3}

ABSTRACT. Suppose one wants to monitor a domain with sensors, each sensing a small ball-shaped region, but the domain is hazardous enough that one cannot control the placement of the sensors. A prohibitively large number of randomly placed sensors could be required to obtain static coverage. Instead, one can use fewer sensors by providing mobile coverage, a generalization of the static setup wherein every possible evader is detected by the moving sensors in a bounded amount of time. Here, we use topology in order to implement algorithms certifying mobile coverage that use only local data to solve the global problem. Our algorithms do not require knowledge of the sensors’ locations, only their connectivity information. We experimentally study the statistics of mobile coverage in two dynamical scenarios. We allow the sensors to move independently (billiard dynamics and Brownian motion), or to locally coordinate their dynamics (collective animal motion models). Our detailed simulations show, for example, that collective motion can enhance performance: The expected time until the mobile sensor network achieves mobile coverage is lower for the D’Orsogna collective motion model than for the billiard motion model. Further, we show that even when the probability of static coverage is low, all possible evaders can nevertheless be detected relatively quickly by the mobile sensor network.

1. INTRODUCTION

Suppose we would like to monitor a given domain with a network of sensors. In certain important situations, we may not be able to prescribe exact locations for the sensors. Fallen rubble might render a domain difficult to access. Monitoring forest fires requires navigating hazardous domains. Further, the sensors may have limited capabilities. We assume that each sensor can detect objects only within a ball of small radius. In such situations, placing the sensors randomly may yield static coverage, meaning that the domain is covered at a single moment in time by the union of the sensing balls. The statistics underlying random static coverage have been extensively studied \cite{19}. Surprisingly, we can use topological techniques to verify when a random placement of sensors produces static coverage, even if the sensor positions are not known! Indeed, the network connectivity of the sensors alone is sufficient input \cite{9,10}.

The number of randomly placed sensors required for static coverage may be prohibitively large. Further, environmental dynamics, such as those produced by wind or ocean currents, may cause sensors to move. We therefore study mobile coverage, a type of dynamic coverage that often requires significantly fewer sensors than its static counterpart. Indeed, we consider mobile sensor networks wherein the sensors may move deterministically or stochastically. By mobile coverage we mean that no possible intruder could continuously move in the domain under surveillance without at some point being seen by at least one of the moving sensors. These sensors may achieve mobile coverage even in the absence of static coverage — i.e. even if the entire domain is never covered at any fixed moment in time by the union of the sensing balls.

\textsuperscript{1}Department of Mathematics, University of Florida, Gainesville, Florida, USA
\textsuperscript{2}Department of Medicine, University of Chicago, Chicago, Illinois, USA
\textsuperscript{3}Department of Mathematics, University of Houston, Houston, Texas, USA

Date: August 12, 2024.

2020 Mathematics Subject Classification. 37M05, 55U10, 68U05.

Key words and phrases. Applied topology, collective motion, evasion path, minimal sensing, mobile coverage, mobile sensor network, pursuit-evasion, random coverage.

*Corresponding author: Contact william.ott.math@gmail.com with inquiries.
Subtle topological tools exist that guarantee mobile coverage, even if the sensors are not able to monitor their position coordinates [9–11]. Using these tools as a foundation, we initiate the study of the statistics of mobile coverage in minimal mobile sensor networks. By minimal, we mean that each sensor measures only a small amount of local information, and in particular the sensors do not measure their locations. As a consequence, we must verify mobile coverage using only local data. This local data consists of connectivity information (overlapping sensors can detect each other), weak angular data (sensors can measure cyclic orderings of neighbors), and distance data (each sensor can measure distances to nearby detected sensors). We address the following statistical questions. Given randomly placed sensors which then move deterministically or stochastically in a domain, how do the statistics of the (random) mobile coverage time $T_{\text{max}}$, the time at which mobile coverage is achieved, behave? Additionally, how do the mean and variance of $T_{\text{max}}$ depend on the sensor dynamics?

We provide concrete algorithms that certify mobile coverage and use them to examine several models of sensor motion; see [1] for code and documentation. These models fall into one of two categories — either the sensors move independently, or they locally coordinate their dynamics. We examine two models wherein the sensors move independently, one deterministic (billiard motion) and one stochastic (Brownian motion). Inspired by the emergent dynamics associated with collective animal motion, we study the seminal D’Orsogna model [12] as well. In this model, sensors use local attractive and repulsive forces to coordinate motion.

As an example application, suppose that deploying a sensor requires a fixed cost, and that having the sensors turned on while roving requires a marginal cost. What is the cheapest way to achieve coverage? Is it cheaper to try and achieve static coverage, by deploying enough randomly-placed sensors to cover the domain at the start time with high probability? Or, is it cheaper to deploy fewer sensors, which likely will not cover the domain initially, but will provide mobile coverage after a certain duration of wandering? How many mobile sensors do we expect to be the most cost-efficient? Our simulations aim to allow us to answer such questions.

We survey related work in Section 2, discuss mobile coverage problems and theoretical tools in Section 3, show the alpha complex can be computed using local information in Section 4, and introduce our algorithms in Sections 5 and 6. We examine how mobile sensor network performance statistics depend on the dynamics of the mobile sensors in Sections 7 and 8. We finish with open questions and concluding remarks in Section 9.

2. Related Work

We discuss related work on static coverage, on a variety of mobile sensor network models, on topological approaches to minimal sensing, on the minimal sensing problem we study in this paper, on Reeb graphs, and on further extensions to the evasion problem.

2.1. Static coverage. What is the probability that immobile ball-shaped sensors cover a domain? This is equivalent to asking what is the probability that mobile ball-shaped sensors cover a domain at time zero. These static coverage questions have a long history in statistics and mathematics. Sections 1.6 and 3.7 of [19] discuss both upper and lower bounds on the probability of coverage by $n$ sensors, and [36] addresses moment problems related to static coverage. Exact formulas for the probability of coverage by a fixed number of sensors are typically unknown. Instead, it is common to take a limit where the number of sensors goes to infinity while their radii go to zero. Kellerer [22] considers the expected number of connected components of a random sample of balls, as well as the expected Euler characteristic. For connections between static coverage and the properties of convex hulls of multivariate samples, see [29].

2.2. A panoply of mobile sensor network models. In this paper, we study a mobile sensor network problem. Before we describe the problem of interest, let us survey a few of the variants on mobile sensor network problems that have been considered.
There are a wide variety of pursuit-evasion mobile sensor network problems. Liu, Dousse, Nain, and Towsley [27] require that the evader remains in a given sensing ball for a certain amount of time before detection occurs. They show in this context that sensors should not move too quickly if we wish to optimize mobile sensor network performance. By contrast, in the evasion problem we will consider, we deem an evader to have been detected at the instant it enters a sensing ball. One could study intelligent sensor networks wherein sensor trajectories actively respond to evader dynamics. For example, [6, 26, 27] study game-theoretic approaches for the optimal evasion and detection strategies in the setting when both evaders and sensors are intelligent. However, in the evasion problem we consider, the sensor motion and evader motion will be independent. One could impose constraints on evader motion (such as bounded speed) and then ask what is the optimal evasion strategy, given these constraints. We will assume that evaders can move arbitrarily fast, provided their motion remains continuous. We refer the reader to [8] for a taxonomy of many different of pursuit-evasion problems.

A variety of ideas have been harnessed to analyze and improve the effectiveness of mobile sensor networks. From a control theory perspective, flocking algorithms have been combined with Kalman filters for efficient target tracking [25, 30–33, 37, 38]. The collective behavior of the ant species *Temnothorax albipennis* has inspired distributed coordination algorithms for tracking [40]. Neural networks can learn a priori unknown environments while dynamic coverage is achieved [35].

2.3. Topological approaches to minimal sensing problems. In minimal sensing, one tries to use sensors measuring only local data in order to solve a global problem. Topological techniques naturally facilitate local-to-global analysis. See [11] for an overview of topological approaches to minimal sensing problems.

Two of the papers introducing coverage problems in sensor networks from the topological perspective are by de Silva and Ghrist. In [9], the authors consider a wide variety of sensor models (static sensors, mobile sensors) and problems (detecting static coverage, detecting mobile coverage, identifying and turning off redundant sensors). They show how coordinate-free connectivity data, when combined with homology, can address this diverse spectrum of problems. In [10], the authors incorporate persistent homology in order to obtain more refined coverage guarantees.

2.4. The evasion problem. The mobile sensor coverage we consider in this paper is derived from Section 11 of [9]. In this section, de Silva and Ghrist derive a one-sided criterion using relative homology, which can guarantee the existence of mobile coverage, but which cannot be used to show the nonexistence thereof. In [3], Adams and Carlsson develop a streaming version of this one-sided criterion using zigzag persistent homology. Zigzag persistence can also be used for hole identification, tracking, and classification in coordinate-free mobile sensor networks [17].

In the static setting, connectivity data alone determines if the region is covered or not [9, 10]. However, as Adams and Carlsson show, time-varying connectivity data alone (and hence zigzag persistence) cannot determine whether or not mobile coverage exists [3]. As a result, Adams and Carlsson augment the connectivity data by considering planar sensors that also measure the cyclic orderings of their overlapping neighbors, as well as distances to these neighbors. This is the type of minimal mobile sensor networks we study in this paper.

2.5. Connections to Reeb graphs. One can model a mobile sensor network as two sets in spacetime: a covered region and an uncovered region. The map to time can be thought of as a real-valued function, whose domain we often restrict to be only the uncovered region. The Reeb graph of the uncovered region, equipped with this real-valued map to time, encodes the space of possible (undetected) intruder motions. From this Reeb graph, it is easy to determine whether or not an evasion path exists. Our algorithms in this paper allow us to not only detect whether or not an evasion path exists, but also to compute the entire Reeb graph of the uncovered region. Though there are a wide variety of algorithms for computing Reeb graphs [4, 34], they typically
focus on computing the Reeb graph of the measured space (the covered region), where in this paper we compute the Reeb graph of the complement of this space (the uncovered region).

2.6. Further extensions to the evasion problem. In a more theoretical direction, the papers [5, 18] give if-and-only-if criteria for the existence of an evasion path using positive homology and Morse theory, but do not address how minimal sensors would measure the inputs required for these criteria. For example, the input required to compute the example in Section 6 of [18] is enough information to instead directly compute the Reeb graph of the uncovered region. The papers [2,3,5] raise and partially address the problem of computing the entire space of evasion paths, which in [24] is implemented for planar sensors that can measure cyclic orderings of and local distances to nearby sensors. Though [24] assumes that each sensor knows its positional coordinates, this assumption can be slightly weakened: if instead each sensor only knows which sensors it overlaps with, and the exact distances to overlapping sensors, then this is enough information to compute alpha complexes (see Section 4).

3. Mobile coverage problems and theoretical tools

Let \( D \subseteq \mathbb{R}^2 \) be a compact domain that is homeomorphic to a closed disk. We assume that the boundary \( \partial D \) is piecewise-smooth. Let \( X \subseteq D \) be a finite set of sensor locations. We equip the sensors with a sensing radius \( r \), within which they can detect intruders and other sensors. More precisely, each sensor \( x \in X \) covers a ball \( B_r(x) = \{ y \in \mathbb{R}^2 : \|y - x\| \leq r \} \). Sensor \( x \) immediately detects any intruder or sensor inside this sensing ball. We further assume that there exists a subset of immobile fence sensors, \( F \subseteq X \), that cover the boundary of the domain. In symbols, we require \( \partial D \subseteq \cup_{x \in F} B_r(x) \). We may think of these fence sensors as being on the boundary, though this is not strictly necessary.

The covered region of the sensors is the set \( C = D \cap (\cup_{x \in X} B_r(x)) \). The static coverage problem asks if \( D = C \) (equivalently, if \( D \setminus C = \emptyset \)). We do not need to know the coordinates of the sensors in order to address this problem. Indeed, using only sensor connectivity data as input, de Silva and Ghrist [9,10] formulate homological conditions that determine static coverage.

3.1. Mobile coverage. This paper focuses on mobile coverage problems, wherein sensors are allowed to move. We modify the static setup as follows. Let \( I = [0,T] \) parametrize time, for some fixed maximum value \( T > 0 \). Each sensor now traces a continuous path \( x : I \to D \). Let \( X \) denote the collection of these paths. As in the static case, we assume there exists a subset of \( X \) corresponding to fence sensors. The fence sensors are immobile, meaning \( x(t) = x(t') \) for all \( t, t' \in I \) if \( x \) is a fence sensor. We continue to assume that the boundary \( \partial D \) is covered by the sensing balls of these (static) fence sensors.

We are interested in the ability of the mobile sensor network to detect all possible continuously moving intruders.

Definition 3.1. The time-varying covered region \( C(t) \) is the union of the sensor balls at time \( t \), namely \( C(t) = D \cap (\cup_{x \in X} B_r(x(t))) \) for \( t \in I \). An evasion path in the mobile sensor network is a continuous path in the uncovered region, i.e. a continuous map \( p : I \to D \) with \( p(t) \notin C(t) \) for all \( t \in I \). We say that a mobile sensor network exhibits mobile coverage on \( D \times I \) if no evasion path exists.

Note that we place no restrictions on the modulus of continuity of an evasion path (we do not assume Hölder or Lipschitz continuity, for instance). In particular, we place no bound on the speed at which a potential evader may travel, although we do require it to move continuously. An intruder successfully avoids detection by moving continuously and never falling into the time-varying covered region. Our problem of mobile coverage is really a question about the existence of evasion paths.

Problem 3.2. Fix \( T > 0 \). Given sensor paths \( (x_n) \), determine if an evasion path exists on \( D \times [0,T] \).
We extend Problem 3.2 by not fixing $T$ in advance, but rather asking for the maximal time interval over which an evasion path exists.

**Problem 3.3.** Given sensor paths $(x_n)$, find the *detection time*

$$T_{\text{max}} = \sup\{T \geq 0 : \text{there exists an evasion path on } D \times [0, T]\}.$$ 

We note that $T_{\text{max}}$ might be infinite.

These mobile coverage problems motivate our two primary goals. First, we want to develop efficient algorithms that solve these mobile coverage problems using as little information as possible about the sensor paths. Our second goal is to understand how the statistics of this detection time depend on sensor dynamics, the number of sensors, the sensing radius, and the geometry of the domain $D$.

Suppose that each sensor can measure the cyclic ordering of its neighbors. If the sensor network remains connected at each time and if the sensors can measure the time-varying alpha complex, then there exists an if-and-only-if criterion for the existence of an evasion path [3, Theorem 3]. The paper [3] also outlines a constructive method for determining the existence of these evasion paths. We now give the necessary background material to make this algorithm explicit.

### 3.2. The alpha complex

Fix a finite set $X$ of points in $\mathbb{R}^2$ (the sensor positions at a fixed time). For any $x \in X$, the *Voronoi region* $V_x \subseteq \mathbb{R}^2$ is defined as

$$V_x = \{ y \in \mathbb{R}^2 : \|y - x\| \leq \|y - x'\| \text{ for all } x' \in X \setminus \{x\} \}.$$ 

That is, the Voronoi region about $x$ contains all points in the plane that are at least as close to $x$ as they are to any other point in $X$. For example, if $X = \{x, x'\}$ consists of two points in the plane, then each Voronoi region will be a half-space, with the perpendicular bisector of the line segment $xx'$ as the line dividing the two half-spaces.

**Figure 1.** Alpha complex produced by a mobile sensor network at a fixed time.

Black stars denote sensor positions and shaded disks depict sensing balls. Line segments and shaded triangles represent 1-simplices and 2-simplices in the alpha complex, respectively.

Each sensor $x \in X$ covers a closed ball $B_r(x)$ of radius $r$. The alpha complex of $X$ is defined as follows [13,15]. Consider the convex regions $B_r(x) \cap V_x$ obtained by intersecting the closed ball of radius $r$ about $x$ with the Voronoi region corresponding to $x$. The *alpha complex* is a simplicial
complex, with vertex set $X$, that contains a $(k - 1)$-simplex if the corresponding $k$ convex regions $B_r(x) \cap V_x$ have a point of mutual intersection; see Figure 1 for an example. In particular, we have a vertex (a 0-simplex) in the alpha complex for each convex region $B_r(x) \cap V_x$, i.e. we have a vertex for each point $x \in X$. We have an edge (a 1-simplex) $xx'$ in the alpha complex whenever two convex regions $B_r(x) \cap V_x$ and $B_r(x') \cap V_{x'}$ intersect. When three such regions share a point of intersection, we have a triangle (a 2-simplex) in the alpha complex. By the nerve lemma [20, Corollary 4G.3], the alpha complex is homotopy equivalent to the union of the convex regions $B_r(x) \cap V_x$, which is the same space as the union of the sensing balls $B_r(x)$ (the covered region).

We prove in Section 4 that the alpha complex can be determined from only the pairwise distances between sensors whose corresponding sensing balls overlap. This is local data — we do not need to know the distances between non-overlapping sensing balls in order to compute the alpha complex. Hence we can use alpha complexes in minimal sensing algorithms.

3.3. Ribbon graphs and boundary cycles. A ribbon graph [21, 28] is a graph equipped with a cyclic ordering of the directed edges about each of its vertices. This information is referred to as rotation data or weak rotation data: it is “weak” in the sense that exact angles are not needed. Using the cyclic ordering, the directed edges can be partitioned into closed loops known as boundary cycles. These boundary cycles will be used to identify connected components of the uncovered region. This translation from cyclic information about vertices to boundary cycles is useful in our study of mobile planar sensor networks for the following reason: cyclic orderings of neighbors can be detected by sensors with minimal capabilities because this information is local in nature. For example, each sensor could use a rotating radar tower to order its neighbors. By contrast, boundary cycles are global in nature, and some of these boundary cycles correspond to holes in the sensor network. Therefore, by using the translation from rotation information to boundary cycles, we are able to convert local data measured by sensors into global data representing the connected components of the uncovered region.

Figure 2. An example ribbon graph (left) along with its boundary cycles (right).

Figure 2 (left) depicts a sample ribbon graph equipped with labels on its vertices and half-edges.\footnote{Using the word “half-edge” instead of “directed edge” connotes that the edge will appear with both orientations.} Each half-edge is oriented. For example, 1 denotes the half-edge from vertex $A$ to $B$, whereas 2 denotes the half-edge from $B$ to $A$. Given a ribbon graph, we have two bijections $\alpha$ and $\sigma$ to and from the set of half-edges on the ribbon graph: $\alpha$ switches the two half edges corresponding to each edge, and $\sigma$ returns the next half-edge about a vertex in a counter-clockwise direction. In Figure 2, for instance, we have $\alpha(1) = 2$ and $\alpha(2) = 1$, $\alpha(3) = 4$ and $\alpha(4) = 3$, et cetera. At vertex $B$, we
have \( \sigma(3) = 20 \), \( \sigma(20) = 2 \), and \( \sigma(2) = 3 \). The boundary cycles of a ribbon graph are the orbits of the composition \( \varphi = \sigma \circ \alpha \). In our example, we compute that \( \varphi(2) = 14 \), \( \varphi(14) = 12 \), \( \varphi(12) = 17 \), \( \varphi(17) = 19 \), and \( \varphi(19) = 2 \), so one of the boundary cycles is the orbit \((2, 14, 12, 17, 19)\). The other three boundary cycles in this example are \((4, 20, 21, 22, 18, 10, 15)\), \((6, 16, 8)\), and \((1, 3, 5, 7, 9, 11, 13)\); see Figure 2 (right).

We note that each boundary cycle corresponds to a connected component of the complement of the graph [3,39]. As a consequence, each boundary cycle corresponds either to a connected component of the complement of the alpha complex, or alternatively to a 2-simplex of the alpha complex. Therefore, the boundary cycles will allow us to identify the Reeb graph of the uncovered region in spacetime.

### 3.4. The Reeb graph

Let \( C(t) \subseteq D \) be the covered region at time \( t \), and let its complement \( U(t) = D \setminus C(t) \) be the uncovered region at time \( t \). In spacetime \( D \times I \), let \( \mathcal{C} = \cup_{t \in I}(C(t) \times \{t\}) \) denote the covered region of spacetime, and let \( \mathcal{U} = \cup_{t \in I}(U(t) \times \{t\}) \) denote the uncovered region of spacetime.

When we use the terms covered region and uncovered region, we are referring to either a subset of space or to a subset of spacetime, depending on the implied context.

Let \( \pi: \mathcal{U} \to I \) be the map from the uncovered region to time, given by \( \pi(x,t) = t \). The Reeb graph of this map \( \pi \) encodes how the connected components of the uncovered region vary with time: how they split, merge, are born, and die. The goal in many mobile evasion problems can be recast as finding the Reeb graph of \( \pi \), since from the Reeb graph, we can read off all possible evasive motions moving forward in time.

![Figure 3. Spacetime representations of two mobile sensor networks.](image.png)

Covered regions are gray and uncovered regions are white, with time increasing from left to right. Each Reeb graph of the projection map \( \pi \) from the uncovered region \( \mathcal{U} \) to time \( I \) appears in blue underneath the spacetime representation.

Formally, we define the Reeb graph as follows. Define an equivalence relation on \( \mathcal{U} \) by declaring \((x,t) \sim (x',t)\) whenever \( x \) and \( x' \) belong to the same connected component of \( \pi^{-1}(t) \). In particular, if \( t \neq t' \) then necessarily \((x,t) \not\sim (x',t')\). The Reeb graph of \( \pi \) is the quotient space \( \mathcal{U}/\sim \). See Figure 3 for two examples.

In a mobile sensor network, each boundary cycle of the 1-skeleton of the alpha complex at time \( t \) corresponds either to a connected component of the uncovered region \( U(t) \), or alternatively to a 2-simplex of the alpha complex (i.e. a region of space covered by sensing domains). In solving Problems 3.2 and 3.3, we will track the boundary cycles as they are born, split, evolve, merge, and die. We will maintain labels on boundary cycles over time, encoding whether or not they may contain an intruder. Unseen intruders can never be present in a boundary cycle corresponding to a 2-simplex, and they may or may not be present in other boundary cycles depending on the
time-evolution of the network. By tracking the boundary cycles, we will actually be tracking the connected components of \( U(t) \) as time \( t \) varies. The algorithms presented in this paper are thus equivalent to computing the Reeb graph of \( \pi : U \to I \), using only information about the covered region \( C = (D \times I) \setminus U \).

4. Local distances determine the alpha complex

Since we wish to use alpha complexes in the minimal mobile sensing algorithms we develop, we must show that the alpha complex can be determined from local data. In this section, we show that the alpha complex can be determined from only the pairwise distances between sensors whose corresponding sensing balls overlap.

Let \( X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^m \) be a collection of \( n \) points in general position. Let \( A(X, r) \) be the alpha complex on \( X \) with radius parameter \( r > 0 \), as defined in Section 3.2 for the particular case \( m = 2 \). The following proposition says that if one knows only the local distances between points in \( X \) (those distances between points whose closed balls of radius \( r \) overlap), then this knowledge is sufficient to compute the alpha complex \( A(X, r) \).

**Proposition 4.1.** Suppose that for each \( i, j \in \{1, \ldots, n\} \) we know
- whether or not \( \|x_i - x_j\| \leq 2r \), and
- the value of \( \|x_i - x_j\| \) if \( \|x_i - x_j\| \leq 2r \).

Then we can determine the alpha complex \( A(X, r) \).

To prove this claim we need some definitions and a lemma from [23] (see also [7] for related ideas). Let \( \sigma \) be a \( k \)-simplex with vertices in \( X \subseteq \mathbb{R}^m \). Due to our general position assumption, there is a unique \( (k-1) \)-dimensional sphere in the \( k \)-plane containing \( \sigma \) that circumscribes \( \sigma \). Let the center and radius of this sphere be the **circumcenter** and **circumradius** of \( \sigma \). Let the \( m \)-dimensional ball with this center and radius be the **circumball** of \( \sigma \).

**Definition 4.2.** A simplex \( \sigma \) is **short** if its circumradius is at most \( r \).

**Definition 4.3.** A simplex \( \sigma \) is **Gabriel** if its circumball has no point of \( X \) in its interior.

**Lemma 4.4** ([23]). A simplex \( \sigma \) is in the alpha complex \( A(X, r) \) if and only if it is
- short and Gabriel, or
- the face of a simplex that is short and Gabriel.

We would like to thank Michael Kerber for showing us the following proof.

**Proof of Proposition 4.1.** Let \( \sigma \) be a \( k \)-simplex with with vertices in \( X \). By Lemma 4.4 it suffices to determine if \( \sigma \) is short and Gabriel, because \( A(X, r) \) is the collection of all such simplices and their faces.

If the distance between any two vertices of \( \sigma \) is more than \( 2r \), then \( \sigma \) is not short. Hence we assume that all pairwise distances between vertices of \( \sigma \) are at most \( 2r \), and so we know these distances exactly. We thus know the shape of \( \sigma \) up to a rigid motion of \( \mathbb{R}^m \). This allows us to determine the circumcenter and circumradius of \( \sigma \). Simplex \( \sigma \) is short if its circumradius is at most \( r \).

Next we must determine if \( \sigma \) is Gabriel. Let \( x \) be a point in \( X \) that is not a vertex of \( \sigma \); we must decide if \( x \) is in the circumball of \( \sigma \). If the distance between \( x \) and any vertex of \( \sigma \) is more than \( 2r \), then \( x \) is not in the circumball of \( \sigma \) because the circumball has radius at most \( r \). Hence we assume that the distance between \( x \) and each vertex of \( \sigma \) is at most \( 2r \), and so we know these distances exactly. We thus know the shape of \( \sigma \cup \{x\} \) up to a rigid motion of \( \mathbb{R}^m \). This allows us to determine if \( x \) is in the circumball of \( \sigma \), and hence if \( \sigma \) is Gabriel. ■
5. Algorithm for mobile coverage

The algorithm we present is based on [3, Theorem 3], which states that the time-varying alpha complex and time-varying rotation information (the cyclic order of edges around each vertex) are sufficient to determine whether or not an evasion path exists in a connected mobile sensor network. While making this theorem algorithmic, one of our key contributions is to present an implementation which is not state-based, i.e. not based on cases.

The structures used in [3, Theorem 3] are presented in an ad hoc way; what is the intuition behind using the alpha complex with rotation data? The intuition comes from [3, Figure 13], which shows that the time-varying connectivity data\(^2\) in a sensor network alone cannot determine whether or not an evasion path exists. More information about how the covered region is embedded into spacetime is required. Since our sensors live in the plane, this extra embedding information can take the form of weak rotation data: each sensor measures the cyclic order of its neighboring sensors. Through the ideas presented in Section 3.3, we are able to obtain a combinatorial representation for each of the uncovered region’s connected components. We note that this rotation information distinguishes the two examples in [3, Figure 13]. The question of whether or not the time-varying Čech complex of a sensor network, in conjunction with rotation data, is enough to determine if an evasion path exists is unknown — see the open question on Page 13 of [3]. However, [3, Theorem 3] shows that the time-varying alpha complex is sufficient when equipped with rotation data. This result relies on the fact that the 1-skeleton of an alpha complex forms a planar graph whereas the 1-skeleton of a Čech complex may not; see [3, Figure 14].

The main idea is to use the rotation data to extract the boundary cycles of this planar graph. The boundary cycles effectively partition the domain into polygonal regions. Some of these regions will correspond to connected components of the complement of the covered region, that is, regions with a portion that is uncovered by the sensors. The other regions will correspond to the 2-simplices of the alpha complex. This allows us to identify some boundary cycles with connected components of the complement of the covered region, and the others with 2-simplices. We can then maintain labels on all boundary cycles and record whether the associated region may contain an intruder or not. Indeed, these labels can be used to recover the Reeb graph, and thus determine whether or not an evasion path exists.

It should be noted that in [3, Theorem 3] the authors assume the alpha complex remains connected for all times \(t\). We will maintain this assumption for now; see Section 6.2 for more on relaxing this assumption.

5.1. Discrete-time problem. So far, the evasion path problem, Reeb graph, and time-varying alpha complex have all been continuously varying in time. We now discretize the problem in time so that we can track evasion paths in a time-stepping fashion. In [3], the authors assume that the times at which changes in the alpha complex occur are known \textit{a priori}, and the time steps used in their algorithm are the time intervals between these changes. If we wish to solve this problem without knowing in advance the times at which changes occur, we need to think about time stepping in a different way. Instead of finding the exact times when the alpha complex changes, as studied in [23], we proceed in a more computationally efficient way.

Instead of stepping from change to change, we will split the time domain \(I\) into discrete time intervals \([t_n, t_{n+1}]\), where \(\Delta t = t_{n+1} - t_n\) is constant. In doing this, we can think of the changes in the alpha complex as happening in the interior of the intervals. If we follow the assumption in [3] that only one of a fixed set of possible changes happens in the alpha complex at any time, then we can choose \(\Delta t\) small enough so that at most one change occurs in each interval. We will continue temporarily on the assumption that \(\Delta t\) is sufficiently small such that any subinterval of \([t_n, t_{n+1}]\)

\(^2\)i.e., either the time-varying Čech, alpha, or Vietoris-Rips complexes
will contain at most one combinatorial change in the alpha complex. In practice, it will generally not be possible to know how small to pick $\Delta t$; we will address this in Section 6.1.

We can now solve a discrete evasion path problem on $[t_n, t_{n+1}]$ using only information from time $t_n$ and $t_{n+1}$. Formally, on each time interval we solve the following time-discrete problem:

**Problem 5.1.** Let $\mathcal{H}_n = \{p(t_n) \in D \mid p \text{ is an evasion path on } D \times [0, t_n]\}$ be the region which may contain an intruder at time $t_n$. Determine if there exists an evasion path $p : [t_n, t_{n+1}] \rightarrow D$ such that $p(t_n) \in \mathcal{H}_n$.

The idea in using an alpha complex on $X$ is to partition $D$ into regions which can be uniquely identified via boundary cycles and will be labelled as feasibly having an intruder or not. If a region is marked as having a possible intruder at times $t_n$ and $t_{n+1}$, then under some modest assumptions we can say there is an evasion path on $[t_n, t_{n+1}]$; see Section 5.2. In this manner, the outline of the algorithm is straightforward and is shown in Algorithm 1. Once the labelling has been determined at time $t_{n+1}$, we can solve Problem 5.1 by simply looking for any regions labelled as feasibly having an intruder. An evasion path will exist on $[t_n, t_{n+1}]$ if any boundary cycle is marked as possibly having an intruder at time $t_{n+1}$. In solving the Problem 5.1, we additionally compute all information necessary to proceed forward in time.

**Algorithm 1: Single Time Step**

**Input**: Alpha complex $A_n$, boundary cycles $B_n$, sensing radius $R$, boundary cycle labelling $L_n : B_n \rightarrow \{0, 1\}$ at $t_n$

**Output**: A mapping $L_{n+1} : B_{n+1} \rightarrow \{0, 1\}$, indicating whether each boundary cycle at $t_{n+1}$ may contain an intruder

begin

$X_{n+1} \leftarrow \text{UpdateSensors}();$

$A_{n+1} \leftarrow \text{ComputeAlphaComplex}(X_{n+1}, R);$

$\ rot\_data \leftarrow \text{ComputeRotationalData}(X_{n+1});$

$B_{n+1} \leftarrow \text{ComputeBoundaryCycles}(A_{n+1}, \ rot\_data);$

$L_{n+1} \leftarrow \text{UpdateBoundaryCycleLabelling}(B_n, B_{n+1}, A_n, A_{n+1}, L_n);$

return $L_{n+1}$

end

Solving this problem on successive time intervals allows us to determine the mobile coverage, and existence of evasion paths, independently from the time domain used. In particular, the end time and times at which changes occur in the alpha complex need not be known a priori. Indeed, the following lemma can be seen via a simple inductive argument:

**Lemma 5.2.** Solving Problem 5.1 iteratively produces a solution to the continuous evasion path Problem 3.2 and the time-discrete analog of Problem 3.3.

5.2. **Combinatorial changes in the alpha complex.** Since the alpha complex is homotopy equivalent to the union of sensor balls, the alpha complex can determine the statically covered region of a sensing network (for further details see [9, 10], which instead use the closely related Vietoris-Rips complex). This allows us to obtain a combinatorial representation of the sensor network’s covered region, allowing us to fully discretize the evasion path problem. One feature of using the alpha complex on $X$ is that because its edges do not cross, it is easy to construct a ribbon graph from it.

Now that we have discretized the problem temporally, we may talk about a discrete sequence of static alpha complexes, each giving a combinatorial representation of the topology of the covered region at time $t_n$. We will denote $A_n$ as the alpha complex on $X$ with radius $r$ at time $t_n$. We will further denote the sets of 1- and 2-simplices of $A_n$ as $A_n^1$ and $A_n^2$, respectively.
The ideas presented in [3] require that only one of a fixed list of \textit{atomic} combinatorial changes in the alpha complex occurs at each point in time. The list of allowed atomic changes can be found in Figure 4. In order to allow more complex combinatorial changes to happen at a time, the ideas in [14] may be helpful. In the discrete case, we require that at most one change happens between \( t_n \) and \( t_{n+1} \); we will assume this for the remainder of Section 5. This can be achieved for instance by assuming that \( \Delta t \) is sufficiently small so that all atomic changes are sufficiently resolved. In this context, \( A_n \) and \( A_{n+1} \) will differ by at most one of these changes.

![Figure 4. Atomic combinatorial changes within the alpha complex. (a) A single edge appears or disappears. (b) A single 2-simplex appears or disappears. (c) A 2-simplex and a free face edge simultaneously appear or disappear. (d) A Delaunay edge flip occurs.](image)

5.3. \textbf{Tracking changes in boundary cycle labels.} Since we are solving this problem in a streaming fashion, we do not know which of the allowed combinatorial changes will occur in the alpha complex. We simply compute \( A_{n+1} \) and the associated boundary cycles and must infer by comparison with the previous time step how the boundary cycle labelling will change. Given that a combinatorial change occurred, we wish to (1) identify which type of combinatorial change occurred, (2) analyze the changes it will induce in the boundary cycles, and (3) determine how the associated boundary cycle labelling will change.

Suppose we know the alpha complexes at time \( t_n \) and \( t_{n+1} \), denoted \( A_n \) and \( A_{n+1} \). Additionally, assume the set of boundary cycles, \( B_n \), are known at time \( t_n \) with the associated labelling \( L_n : B_n \rightarrow \{0, 1\} \). We let 0 denote that the corresponding cycle cannot contain an intruder, and 1 denote that the corresponding cycle may contain an intruder. From \( A_{n+1} \) we can compute the rotation data and obtain the associated boundary cycles \( B_{n+1} \). Now with this information as input, we want to determine which combinatorial change occurred, understand the associated changes in boundary cycles, and produce the appropriate boundary cycle labelling \( L_{n+1} : B_{n+1} \rightarrow \{0, 1\} \).

We can determine the transition that occurred by counting the number of boundary cycles, 1-simplices, and 2-simplices that are new and those that are no longer present. For example, when an edge is added, this can be uniquely detected by verifying that

\[
\begin{align*}
|A_{n+1}^1 \setminus A_n^1| &= 1, & |A_n^1 \setminus A_{n+1}^1| &= 0, \\
|A_{n+1}^2 \setminus A_n^2| &= 0, & |A_n^2 \setminus A_{n+1}^2| &= 0, \\
|B_{n+1} \setminus B_n| &= 2, & |B_n \setminus B_{n+1}| &= 1
\end{align*}
\]
Since the network is connected, this new edge necessarily splits a boundary at time \( n \) into two new ones at time \( n + 1 \). A table for each atomic transition is given in Table 1.

| Action                                | Edges (+) | Edges (−) | 2-simplices (+) | 2-simplices (−) | Boundary cycles (+) | Boundary cycles (−) |
|---------------------------------------|-----------|-----------|-----------------|-----------------|---------------------|---------------------|
| Add edge                              | 1         | 0         | 0               | 0               | 2                   | 1                   |
| Remove edge                           | 0         | 1         | 0               | 0               | 1                   | 2                   |
| Add 2-simplex                         | 0         | 0         | 1               | 0               | 0                   | 0                   |
| Remove 2-simplex                      | 0         | 0         | 0               | 1               | 0                   | 0                   |
| Add free edge and 2-simplex           | 1         | 0         | 1               | 0               | 2                   | 1                   |
| Remove free edge and 2-simplex        | 0         | 1         | 0               | 1               | 1                   | 2                   |
| Delaunay edge flip                    | 1         | 1         | 2               | 2               | 2                   | 2                   |

Table 1. The number of simplices and boundary cycles added (+) or removed (−) under each atomic transition.

We now analyze how each atomic transition affects the boundary cycles and their labelling; see Figures 4 and 5 for illustrations of each atomic transition.

(a+) **1-simplex is added.** When a 1-simplex is added, there will always be one old boundary cycle being split into two new boundary cycles. So the labelling of the two new boundary cycles will inherit the label of the old (removed) boundary cycle.

(a−) **1-simplex is removed.** When a 1-simplex is removed, we will always have two old boundary cycles joining together into one new boundary cycle. So the new boundary cycle will be labelled as potentially having an intruder if either of the old boundary cycles may have had an intruder.

(b+) **2-simplex added.** When a 2-simplex is added, once the corresponding boundary cycle is identified, it can be set as having no intruder.

(b−) **2-simplex removed.** When a 2-simplex is removed, the corresponding boundary cycle, which was previously labelled with no intruder, will continue to maintain the same label.

(c+) **Pair consisting of a 2-simplex and a free edge is added.** The label updating is similar to adding a 1-simplex, but now we must determine which boundary cycle corresponds to the new 2-simplex and set its label to have no intruder. This update can be achieved through the composition of (a+) adding a 1-simplex, and then (b+) adding the 2-simplex.

(c−) **Pair consisting of a 2-simplex and a free edge is removed.** Just as in adding a simplex pair, the labelling for this case can be done by (a−) removing the 1-simplex and then (b−) removing the 2-simplex.

(d) **Delaunay edge flip.** A Delaunay edge flip occurs when the two triangles filling in a quadrilateral flip. The new boundary cycles will be labelled false since they are filled-in 2-simplices.

We remark that a Delaunay edge flip occurs only in the setting when four sensor locations lie on the boundary of a single circle, with no sensors in the interior of that circle. Though this occurrence is not generic for static sensors, it will happen for mobile sensors. The Voronoi regions of these four locations have a common intersection point at the center of this circumcircle. The Delaunay edge flip swaps which diagonal edge is present. The presence of such a diagonal edge in the alpha complex implies that the sensing radius is at least as large as the circumradius of these four sensor locations. This implies that the old diagonal edge (prior to the Delaunay edge flip) is adjacent to two 2-simplices in the old alpha complex, and also that the new diagonal edge (after the Delaunay edge flip) is adjacent to two 2-simplices in the new alpha complex.

Each of these atomic changes can also be interpreted in terms of how it affects the Reeb graph. The case (a+) corresponds to a boundary cycle (and thus a connected component of \( U(t) \)) being split into two boundary cycles, and can be associated with a critical point in the Reeb graph. Case (a−) likewise corresponds to two boundary cycles merging together, again corresponding to a critical point in the Reeb graph. Cases (b+) and (b−) imply the death and birth of a boundary...
cycle respectively. Cases (c+) and (c−) do not correspond to a critical point in the Reeb graph since they do not create or destroy any connected components of the complement. However, they do change the representation of the boundary cycles, thus we can associate the old boundary cycle representation with the new boundary cycle representation as belonging to the same connected component of \( U(t) \). Finally, case (d) does not affect the connected components of \( U(t) \), and does not correspond to a critical point in the Reeb graph. Note that the assumption that these atomic changes are the only changes implies that the Reeb graph is a binary graph.

One may explicitly construct a discrete representation of the Reeb graph by associating the times at which each atomic change occurs with a graph node. For each boundary cycle, a directed edge is then inserted between the graph nodes beginning with the time the boundary cycle was produced and terminating at the time the boundary cycle is next modified.

![Figure 5. Atomic boundary cycle updates. These updates correspond to the atomic combinatorial changes within the alpha complex illustrated in Figure 4.](image)

5.4. Non-state-based algorithm. In [3,24], the boundary cycle label update is treated as a state-based algorithm, updating the labeling depending on the case (a+), (a−), (b+), (b−), (c+), (c−), or (d) of which atomic transition occurred. We observe here that, given all the current assumptions, this is not necessary. This was initially surprising to us, given that the proof of [3, Theorem 3] is heavily reliant on its case-based organization. Indeed, this non-state-based update comes from the realization that in each atomic transition we can first add/remove a 1-simplex (if applicable) and then label all 2-simplices as having no intruder while leaving all other labels unchanged. This is put into pseudocode in Algorithm 2, where \( \lor \) denotes the “or” operation. We emphasize that Algorithm 2 is not divided into cases (which for purposes of comparison, would produce the much longer Algorithm 4).

5.5. Where is the fence used? The astute reader (or an anonymous referee who we would like to acknowledge) may notice that the fence sensors do not play a larger role than any other sensors in the algorithm above. In some sense, the fence sensors are not needed. Suppose that the sensors and intruders live in \( \mathbb{R}^2 \). With only a finite number of sensors with finite radii, there will always be a single unbounded connected component of the uncovered region. If this unbounded component contains intruders, then it can never be swept by a finite number of sensors with finite radii. However, suppose that at the initial time, only some of the connected components of the uncovered region contain intruders, and the unbounded component of the uncovered region does
Algorithm 2: Labelling Update - Serialized

Input: Alpha complex $A_{n+1}$, the boundary cycles $B_n$ and $B_{n+1}$, boundary cycle labelling $L_n: \{0, 1\}$

Output: The mapping $L_{n+1}: \{0, 1\}$, indicating whether each boundary cycle may contain an intruder

begin
  for $b \in B_n \cap B_{n+1}$ do \\
    $L_{n+1}[b] \leftarrow L_n[b]$; \\
  end
  for $b \in B_{n+1} \setminus B_n$ do \\
    $L_{n+1}[b] \leftarrow \bigvee_{x \in B_n \setminus B_{n+1}} L_n[x]$; \\
  end
  for $b \in B_{n+1}$ do \\
    if Nodes of $b$ form a 2-simplex of $A_{n+1}$ then \\
      $L_{n+1}[b] \leftarrow \text{False}$; \\
    end
  end

return $L_{n+1}$;

end

not. One could then run our algorithm, maintaining labels to denote which components could contain intruders at each time, to decide in an if-and-only-if fashion whether an evasion path exists or not. As remarked above, if the unbounded component ever merges with one that contains intruders, then the unbounded component will forevermore contain intruders — but that might never happen. (Alternatively, one could also run a version of our algorithm for sensors on the 2-sphere.)

6. Realization of the evasion path algorithm

There are some technicalities that must be overcome if one is to implement the algorithm described in Section 5. Specifically, we have found the assumptions of being able to choose a small enough uniform time step size, as well as maintaining connectivity of the alpha complex, quite strong assumptions if one wishes to use arbitrary models of motion. In this section, we relax these assumptions and provide a way to solve the evasion path problem without changing the nature of the original algorithm.

6.1. Adaptive time stepping. As mentioned above in Section 5.1, we have been assuming that our time step $\Delta t$ is sufficiently small to resolve all combinatorial changes in the alpha complex. If we wish to select $\Delta t$ as a parameter, we have the problem that there is also the underlying inability to know exactly how small is “sufficiently small.” Such knowledge would require that we know at what exact times the alpha complex will undergo atomic changes, which may be possible for simple trajectories [23], and impractical in more complex situations. We found that for this assumption to hold, $\Delta t$ must generally be quite small. With the alpha complex being expensive to compute, this renders time stepping using a uniform $\Delta t$ impractical.

Our solution to this problem is to use an adaptive time-stepping scheme to resolve interval steps of size $\Delta t$ into subintervals of unequal size, each of which contain at most one atomic change. We recall that atomic changes are assumed to happen in the interior of our time intervals. Note that we still require the assumption that at each point in time only one atomic transition occurs. With this in mind, if $\Delta t$ is not small enough, two or more atomic changes may happen within a single
interval (though at different times). If we detect that \( A_n \) and \( A_{n+1} \) differ by more than an atomic change, we can recursively divide the interval in two and solve Problem 5.1 on the subintervals \([t_n, t_n + \frac{\Delta t}{2}]\) and \([t_n + \frac{\Delta t}{2}, t_n + \Delta t]\). This is equivalent to performing a binary search for the intervals which differ by only one atomic change. Pseudocode for this process is shown in Algorithm 3 in the appendix. This algorithm works because Lemma 5.2 provides that if there is an evasion path which differ by only one atomic change. Pseudocode for this process is shown in Algorithm 3 in the appendix. This algorithm works because Lemma 5.2 provides that if there is an evasion path on \([t_n, t_n + \frac{\Delta t}{2}]\) and on \([t_n + \frac{\Delta t}{2}, t_n + \Delta t]\) (overlapping on the same connected component of the uncovered region at time \(t_n + \frac{\Delta t}{2}\)), then there is an evasion path on \([t_n, t_n + \Delta t]\).

Note that if \( A_n \) and \( A_{n+1} \) are identical, this does not imply the same for every subinterval of \([t_n, t_{n+1}]\). For example, removing an edge and then adding the same edge back in within a single time interval could produce identical alpha complexes \( A_n \) and \( A_{n+1} \). Similarly, if \( A_n \) and \( A_{n+1} \) differ by at most one atomic change, this does not imply the same for every subinterval of \([t_n, t_{n+1}]\). For example, if two edges were added and one of the two then removed within a single time interval, \( A_n \) and \( A_{n+1} \) would differ by a single added edge. Since our adaptive time-stepping scheme produces a sequence of alpha complexes such that each consecutive pair is identical or differs by a single atomic change, we assume that if a time step contains at most one atomic change, further refinement will not produce any other atomic changes.

This adaptive time-stepping approach has been developed to minimize the number of alpha complex computations needed to resolve a given interval. Observe that when attempting to resolve \( A_n \) and \( A_{n+1} \), the alpha complex \( A_{n+1} \) can be stored for later reuse when resolving \( A_{n+1-2-i} \) and \( A_{n+1} \) and so on, recursively. The most common approach to computing the alpha complex is to compute the Delaunay triangulation and then filter out unneeded simplices. This approach suffers from the high computational cost of the Delaunay triangulation, namely a complexity of \( O(K^{d+1}) \), where \( K \) is the number of vertices in the alpha complex [16]. If the rotation information can be queried from the sensors directly or maintained dynamically, then this does not add to the complexity. Generation of the boundary cycles has complexity \( O(E) \), where \( E \) is the number of edges in the alpha complex, since we are simply generating the orbits of \( \varphi \). Thus each recursion level has complexity \( O(K^2 + E) \), so the overall algorithm has complexity \( O(M(K^2 + E)) \), where \( M \) is the number of steps produced by the recursion process.

6.2. **Extension to disconnected sensor networks.** In Section 5.3, we operate entirely under the assumption that the sensor network, or equivalently the alpha complex, remains connected. This is a rather strong assumption if we wish to investigate arbitrary models of motion. We need to allow the sensor network to become disconnected.

The algorithm from Section 5, however, does not work in the context of disconnected sensor networks. As can be seen from Figure 16 of [3], a number of problems arise when we attempt to allow for a disconnected alpha complex. Indeed, this figure shows two sensor networks — both of which become disconnected — which have the same alpha complex and rotation information at each time, yet one contains an evasion path while the other does not. Allowing for a disconnected alpha complex creates the following problems, among others: (1) The embedding of the alpha complex in \( \mathbb{R}^2 \) may be ambiguous; (2) the identification of boundary cycles with connected components of the uncovered region is not injective; (3) the boundary cycles must be aware of their embedding relative to each other. Even worse, if we want to fix issue (3), local information is not sufficient; nonlocal information is required.

We now present a modification of the evasion path problem that allows us to determine existence of evasion paths without this rather restrictive assumption of the sensors maintaining connectivity. The main idea is to ignore sensors that become disconnected from the fence component until they are reconnected, in a similar model to that considered in [9]. This can be physically motivated by considering the scenario where disconnected sensors are out of range of the main computing network, i.e., we can imagine a setup where sensors that are disconnected from the fence are disconnected from a power source. For this reason we will refer to this model of allowing disconnected sensors
as the “power-down model.” Analogously, any algorithm which solves Problem 3.2 and allows disconnected sensors to remain turned on will be referred to as a “power-on model.” We do not consider a power-on model here, and instead save it for future work, since such a model requires nonlocal information and would change the nature of the problem being solved.

Topologically, the power-down model changes the definition of the covered region as follows. Recall that \( F \subseteq X \) is the subset of fence sensors. At each time \( t \), we partition the covered region \( C(t) = \bigcup_{x \in X} B_r(x(t)) \) into two regions: the part that is in the same connected component as the fence sensors, and the part that is not in this connected component. This partitions the set \( X \) of sensors into two groups:

\[
X = X' \sqcup X'',
\]

where \( X' \supseteq F \) is the set of sensors in the same connected component of \( C(t) \) as the fence sensors, and where \( X'' = X \setminus X' \) is the set of sensors which are not in this component. In the power-down model, all of the sensors in \( X'' \) are turned off, and do not contribute to the covered region. Indeed, the covered region in the power-down model is \( \tilde{C}(t) = \bigcup_{x \in X'(t)} B_r(x(t)) \). We are still interested in whether or not there exists an evasion path, i.e. a continuous path \( p : I \to D \) with \( p(t) \notin \tilde{C}(t) \) for all times \( t \). With this modified notion of an evasion path, we can proceed with solving Problem 3.2.

We remark that our algorithm for detecting mobile coverage with possibly disconnected sensor networks in the power-down model is still local, relying only on alpha complexes (computed locally as in Section 4) and weak rotation data. Moreover, our local algorithm for the power-down model still gives one-sided bounds for a would-be power-on model: if no evasion path exists with the power-down sensors, then no evasion path exists for power-on sensors undergoing the same motions. Similarly, if an evasion path \( p \) exists for the power-on model, then \( p \) is also an evasion path for the power-down model. These are consequences of the inclusion \( \tilde{C}(t) \subseteq C(t) \) of the covered regions in the two different models, for all \( t \).

An attractive feature of the power-down modification is that it adds no complexity or data structures to existing algorithms. In the context of the previous sections, the only modification is that we allow for two additional atomic combinatorial changes corresponding to components becoming connected or disconnected from the fence. We additionally can ignore all transitions that occur within a disconnected component. We follow the same time discretization process as in Section 5; the only difference is that we must handle the additional cases where components of the alpha complex become disconnected or reconnected. This is a only a minor modification thanks to the easily seen observation:

**Lemma 6.1.** Under the assumption that at most one combinatorial change listed in Section 5.3 happens within each time, then a component of the alpha complex can only become disconnected (or reconnected) if a single edge is removed (or added) to the alpha complex.

### 6.3. Boundary cycle label updates.

When a component of the alpha complex becomes disconnected from the fence, we will have a single boundary cycle initially surrounding a hole which will then be split into two boundary cycles; see Figure 6 for illustration. One of these boundary cycles will correspond to the outside of a new disconnected component; it will not be given any label. The other boundary cycle will be a hole enclosing the disconnected component, and if it is connected to the fence then we will maintain a label for this boundary cycle, and its initial label will be the same as the prior boundary cycle that split. In later time steps, any boundary cycle on a disconnected component will not be labeled. This represents the idea that the sensors that become disconnected are not a part of the power-down alpha complex. This type of atomic change can be uniquely identified just as in Section 5.3; see Table 2.

When a disconnected component becomes reattached to the fence through the addition of a 1-simplex, we will have the two previously described boundary cycles merging into one. All of the newly connected boundary cycles will be labeled to match the outer enclosing boundary cycle; however, any boundary cycle filled by a 2-simplex will be labeled as having no intruder. Again, this
atomic change can be detected by simply counting the number of boundary cycles and simplices in
the alpha complex before and after the transition; see Table 2.

We can detect if a component has become disconnected or reconnected by simply extending Table 1 with the following two cases.

|                     | Edges (+) | Edges (-) | 2-simplices (+) | 2-simplices (-) | Boundary cycles (+) | Boundary cycles (-) |
|---------------------|-----------|-----------|-----------------|-----------------|---------------------|---------------------|
| Disconnect          | 0         | 1         | 0               | 0               | 2                   | 1                   |
| Reconnect           | 1         | 0         | 0               | 0               | 1                   | 2                   |

Table 2. The number of simplices and boundary cycles added (+) or removed (−) under the additional possible atomic transitions (extending Table 1) in the power-down model.

Figure 6. A component of the alpha complex disconnects from the fence component. From left to right, a single edge disappears, producing an additional connected component. From right to left, a single edge appears, merging two connected components into one.

7. Detection time statistics for diffusive sensors

In this section, we compute and examine the detection time $T_{\text{max}}$ distributions assuming the mobile sensors move diffusively. Indeed, diffusion is relevant when minimal sensor networks experience random environmental forces, such as winds.

In all of the models of sensor motion we consider (diffusion, billiards, and collective motion) in this section and the next, the initial locations of the mobile sensors at time zero are chosen uniformly at random inside the domain. We note that $T_{\text{max}}$ is a random variable that can attain the value infinity. (This happens if the sensor network never provides mobile coverage.) Experimentally, we appear to be in regimes where $T_{\text{max}} = \infty$ happens with probability zero, and the expected value $E[T_{\text{max}}]$ is finite. If needed (say for computational purposes), one could replace $T_{\text{max}}$ with a “capped value” that is the minimum of $T_{\text{max}}$ and some large positive constant. But this was not needed for our experiments—all of our trials yielded mobile coverage in finite time.

7.1. Domain geometry. Our domain is a scaled square $D = (1 + \delta)S$, where $S$ is the unit square centered at the origin and $\delta > 0$ is small and fixed. The sensing domains of the immobile fence sensors cover the boundary $\partial D$. We require that the mobile sensors remain within the unit square $S$. In particular, mobile sensors reflect elastically off of the boundary of $S$. We initialize the locations of the mobile sensors by placing them uniformly at random within $S$. 
7.2. Performance of Brownian mobile sensor networks. We assume that each mobile sensor moves independently and satisfies the stochastic differential equation
\[ \text{d}x_t = f(x_t) \text{d}t + \sigma \text{d}W_t, \]
where \( f \) denotes the drift and \( W_t \) denotes standard two-dimensional Brownian motion. We examine \( T_{\text{max}} \) statistics assuming no drift (\( f \) vanishes), upon varying both the sensing radius \( r \) and the number of mobile sensors \( N \). We generated \( T_{\text{max}} \) distributions by performing 500 mobile sensor network simulations for each \((N,r)\) pair and using our algorithms to compute \( T_{\text{max}} \) for each simulation. We used diffusion coefficient \( \sigma = 0.5 \) for every simulation. See Figure 7 for visualization of a sample sensor network initialization for various \((N,r)\) pairs.

Figure 7. Sample sensor network initialization for various \((N,r)\) pairs.

Figure 8d illustrates several of these distributions for \( T_{\text{max}} \). Notice that they are unimodal. Further, the \( T_{\text{max}} \) distribution concentrates near \( t = 0 \) as \( N \) increases (for fixed \( r \)) and as \( r \) increases (for fixed \( N \)). It would be interesting to launch an in-depth mathematical and empirical study of \( T_{\text{max}} \) distributions for the motion models we here consider. Mathematically, one could attempt to prove limit theorems for \( T_{\text{max}} \), for instance when \( N \to \infty \) and \( r \to 0 \) with \( Nr^2 \) held constant. Empirically, one could fit known distributions to the data obtained from simulations in an effort to test conjectures about the distribution of \( T_{\text{max}} \). We make one such conjecture here. For Brownian mobile sensor networks, \( T_{\text{max}} \) (when discretized) is distributed approximately geometrically when \( N \) and \( r \) are large. The distributions for \((N,r) = (35, 0.16)\) and \((N,r) = (25, 0.20)\) shown in Figure 8d support this conjecture.

The probability that the sensing balls collectively cover the entire domain \( D \) at time \( t = 0 \) (that is, the probability of static coverage) varies widely over \((N,r)\)-space (Figure 8c). Importantly,
Figure 8. Detection time statistics for Brownian mobile sensor networks. The heatmaps illustrate (a) mean detection time $E[T_{\text{max}}]$, (b) detection time variance $\text{Var}[T_{\text{max}}]$, and (c) the probability that the union of the sensing balls covers the entire domain $D$ at time $t = 0$. We vary the sensing radius $r$ and the number $N$ of mobile sensors. (d) Detection time distributions (smoothed histograms) for various $(N, r)$ pairs.

Even when the probability of static coverage is low (Figure 8c, lower-left corner), the mobile sensor network nevertheless detects all possible evaders relatively quickly (Figure 8a, lower-left corner).

As expected, Figure 8a indicates that $E[T_{\text{max}}]$ decreases as $r$ increases (for fixed $N$) and as $N$ increases (for fixed $r$). For Brownian mobile sensor networks, $\text{Var}[T_{\text{max}}]$ behaves the same way (Figure 8b).
8. **Collective motion improves mobile sensor network performance**

Suppose the mobile sensors are capable of self-propulsion. How should the mobile sensors move in order to optimize performance of the sensor network? Suppose that each mobile sensor can compute the relative positions of nearby sensors. Can improved mobile sensor network performance emerge when the mobile sensors use this local position data to locally coordinate?

We answer this question affirmatively by allowing the mobile sensors to use attractive and repulsive forces for local coordination. In particular, we use the D’Orsogna model [12], a seminal effort to explain the emergent phenomena that arise when agents move collectively. We compare this ‘locally strategic’ model to a ‘locally oblivious’ alternative wherein the mobile sensors move independently as simple billiard particles. Both the billiard and D’Orsogna models proceed deterministically after the initial random sensor locations and headings have been chosen, in contrast with the stochastic Brownian motion in the previous section.

8.1. **Billiard motion.** Our domain geometry is as in Section 7.1. We initialize the sensor network by placing each of the \( N \) mobile sensors uniformly at random within the unit square \( S \) and assigning each a random initial direction of motion \( \theta_k \in \mathbb{R}/2\pi\mathbb{Z} \). Each mobile sensor moves independently of the others at unit speed and changes direction of motion only when colliding elastically with the virtual boundary \( \partial S \).

8.2. **Collective mobile sensor motion.** The D’Orsogna model uses attractive and repulsive potentials to model the dynamics of animal collectives such as fish schools and bird flocks. Here, we show that if mobile sensors use such potentials to locally coordinate, the emergent collective motion can cause these D’Orsogna sensor networks to outperform billiard sensor networks.

The \( N \) mobile sensors \( x_k \) obey the nonlinear system

\[
\begin{align*}
\frac{dx_k}{dt} &= v_k \\
\frac{d}{dt} v_k &= (\alpha - \beta \|v_k\|^2) v_k - \sum_{m=1}^{N} g(\|x_m - x_k\|) \nabla_k U_m(x_k).
\end{align*}
\]

The generalized Morse potential

\[
U_m(x_k) = C_r e^{-\|x_m - x_k\|/\ell_r} - C_a e^{-\|x_m - x_k\|/\ell_a}
\]

describes the interaction between sensors \( m \) and \( k \). The parameters \( \ell_a \) and \( \ell_r \) in Eq. (2) quantify the length scales for attraction and repulsion, respectively. The parameters \( C_a \) and \( C_r \) quantify attraction and repulsion magnitudes. Constant \( m \) is a sensor’s mass.

The function \( g \) localizes the interaction force to ensure that sensors \( m \) and \( k \) only interact if they are near one another. Our use of the localizer \( g \) is consistent with our minimal sensing approach: The mobile D’Orsogna sensors should only have the ability to gather and react to local conditions. Concretely, \( g \) is given by

\[
g(x) = \begin{cases} 
1, & \text{if } x < 2r, \\
0, & \text{if } x \geq 2r.
\end{cases}
\]

We note that two sensors interact, affecting each other’s motion, if and only if their sensing balls of radius \( r \) overlap.

In isolation, mobile D’Orsogna sensor \( k \) only feels the force \((\alpha - \beta \|v_k\|^2)v_k\) and will therefore approach the asymptotic speed \( \sqrt{\alpha/\beta} \). In equilibrium, speeds of the mobile D’Orsogna sensors should statistically fluctuate about this asymptotic value.
8.3. **Parameter selection and initial data for model comparison.** The mobile billiard sensors move at unit speed. We have set \( \alpha = \beta = 1 \) in Eq. (1b) so that the equilibrium speeds of the mobile D’Orsogna sensors fluctuate about one. This choice matches the characteristic speeds of the two models, allowing for a meaningful sensor network performance comparison. We initialize the billiard and D’Orsogna sensor networks in the same way: We place each of the \( N \) mobile sensors uniformly at random within the unit square \( S \) and we assign each a random initial direction of motion \( \theta_k \in \mathbb{R}/2\pi\mathbb{Z} \).

We have set the attraction and repulsion length scales in the D’Orsogna potential (2) to \( \ell_a = 1 \) and \( \ell_r = 0.1 \), respectively. The scale separation here models collective animal behavior, namely short-range repulsion, and attraction over longer distances. We have set the attraction and repulsion magnitude parameters to \( C_a = 0.45 \) and \( C_r = 0.5 \), respectively. The mobile D’Orsogna sensors have mass \( m = 1 \).

8.4. **D’Orsogna sensor networks outperform billiard sensor networks.** We computed detection time \( T_{\text{max}} \) distributions for both the D’Orsogna sensor network and the billiard sensor network simulations. We did so by performing 500 simulations for each choice of sensing radius \( r \) and number of mobile sensors \( N \). Figure 9 compares \( E[T_{\text{max}}] \) and \( \text{Var}[T_{\text{max}}] \) behavior for the two sensor network models.

Importantly, the D’Orsogna network detects evaders more quickly on average than the billiard network (Figure 9ac). Quantitatively, averaging
\[
\frac{E[T_{\text{max}}](\text{billiard}) - E[T_{\text{max}}](\text{D’Orsogna})}{E[T_{\text{max}}](\text{billiard})}
\]
over all \((N, r)\) pairs, we find that the D’Orsogna network outperforms the billiard network by 23.5%. We speculate that both the repulsive and attractive potentials help the D’Orsogna network outperform the billiard network. The repulsive potential may cause the typical fraction of the domain covered by the mobile D’Orsogna sensors at a fixed time to eclipse that of the mobile billiard sensors. The attractive potential may help prevent individual mobile D’Orsogna sensors (or small groups) from disconnecting from the fence component.

As is the case for the Brownian network, \( \text{Var}[T_{\text{max}}] \) varies monotonically with \( r \) and with \( N \) for the billiard network (Figure 9b) and the D’Orsogna network (Figure 9d).

9. **Conclusion**

Minimal sensor networks attempt to solve global sensing problems using local data. Assuming mobile planar sensors collect local connectivity data, local distance data, and weak rotation information, can we determine whether or not an evasion path exists? Adams and Carlsson answer this question affirmatively [3, Theorem 3]. Here, we have converted this theorem into explicit algorithms that allow us to compute detection time distributions for both deterministic and stochastic mobile sensor networks. We have computed and analyzed these distributions for sensor networks with Brownian, billiard, and D’Orsogna motion models. Comparing the two deterministic mobile networks, the D’Orsogna system outperforms its billiard counterpart. This suggests that emergent dynamics resulting from collective local motion can positively impact mobile sensor network performance.

We end with open questions that we hope will help inspire future research. Opportunities exist for both theoretical and computational investigations.

(1) The expected time to achieve mobile coverage decreases as we increase the number of sensors or as we increase the sensing radius. Do phase transition phenomena ever arise?

(2) In the limit as the number of sensors goes to infinity and the sensing radius of each sensor goes to zero, can one describe the asymptotics of the expected time to achieve mobile coverage?
Figure 9. Performance comparison of D’Orsogna and billiard mobile sensor networks. Heatmaps illustrate detection time statistics as functions of the sensing radius $r$ and the number of mobile sensors $N$. Left column: Mean detection time. Right column: Detection time variance. Top row: Billiard sensor network. Bottom row: D’Orsogna sensor network.

Does the distribution of this detection time, $T_{\text{max}}$, converge in this limit? If it does, to what limiting distribution? How do these asymptotics depend on the model of sensor motion?

(3) For deterministic sensor motion models, how do the statistics of $T_{\text{max}}$ depend on the underlying invariant measure and rate of decay of correlations?

(4) The detection time distributions in Figure 8d are unimodal. Is this always the case for Brownian motion? What happens for different models of motion?
What collective motion models, wherein each mobile sensor can adjust its motion based on the relative positions of its local (but not global) neighbors, are the most effective at decreasing the expected time until mobile coverage is achieved?

What can be said about collections of sensors with randomly varying sensing radii?

What do the statistics for mobile coverage look like in the power-on model, wherein mobile sensors are allowed to become disconnected but still all remain on?

How do the statistics for mobile coverage change as the domain varies? For instance, what happens on a torus, or on other domains that are not simply connected?

How would the presence of hyperbolicity in the dynamics impact mobile sensor network performance? For instance, what happens if the mobile sensors move as billiard particles and reflect off of convex scatter (the Lorenz gas or Sinai billiard setup)?

Suppose we have a fixed domain, say a unit cube or a unit ball in $\mathbb{R}^d$. Let $V > 0$ be a fixed volume that will be distributed amongst sensors. For any integer $N \geq 1$, suppose we have $N$ ball-shaped sensors, each of volume $V/N$, which move in the domain according to (say) Brownian motion. How does the expected time until mobile coverage depend on $N$? Which value of $N$ (as a function of $V$) minimizes the expected mobile coverage time?

Acknowledgments

We would like to acknowledge Michael Kerber for showing us the result in Section 4. We would like to thank the anonymous reviewers for their insightful and helpful feedback.

Funding

This work has been partially supported by National Science Foundation grant DMS 1816315 (William Ott).

Code availability

See [1] for code and documentation.

References

[1] https://github.com/elykwilliams/EvasionPaths.

[2] H. Adams, Evasion paths in mobile sensor networks, PhD thesis, Stanford University, 2013.

[3] H. Adams and G. Carlsson, Evasion paths in mobile sensor networks, International Journal of Robotics Research, 34 (2015), pp. 90–104, https://doi.org/10.1177/0278364914548051.

[4] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno, Reeb graphs for shape analysis and applications, Theoretical computer science, 392 (2008), pp. 5–22.

[5] G. Carlsson and B. Filippenko, The space of sections of a smooth function, 2020, https://arxiv.org/abs/2006.12023.

[6] J.-C. Chin, Y. Dong, W.-K. Hon, C.-T. Ma, and D. Yau, Detection of intelligent mobile target in a mobile sensor network, IEEE/ACM Transactions on Networking, 18 (2010), pp. 41–52, https://doi.org/10.1109/TNET.2009.2024309.

[7] H. Chintakunta and H. Krim, Distributed boundary tracking using alpha and Delaunay–Čech shapes, arXiv preprint arXiv:1302.3982, (2013).

[8] T. Chung, G. Hollinger, and V. Isler, Search and pursuit-evasion in mobile robotics A survey, Autonomous Robots, 31 (2011), pp. 299–316, https://doi.org/10.1007/s10514-011-9241-4.

[9] V. De Silva and R. Ghrist, Coordinate-free coverage in sensor networks with controlled boundaries via homology, International Journal of Robotics Research, 25 (2006), pp. 1205–1222, https://doi.org/10.1177/0278364906072252.

[10] V. De Silva and R. Ghrist, Coverage in sensor networks via persistent homology, Algebraic and Geometric Topology, 7 (2007), pp. 339–358, https://doi.org/10.2140/agt.2007.7.339.

[11] V. De Silva and R. Ghrist, Homological sensor networks, Notices of the American Mathematical Society, 54 (2007), pp. 10–17.
[12] M. D’Orsogna, Y. Chuang, A. Bertozzi, and L. Chayes, Self-propelled particles with soft-core interactions: Patterns, stability, and collapse, Physical Review Letters, 96 (2006), https://doi.org/10.1103/PhysRevLett.96.104302.

[13] H. Edelsbrunner and J. L. Harer, Computational Topology: An Introduction, American Mathematical Society, Providence, 2010.

[14] H. Edelsbrunner and E. P. Mücke, Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms, ACM Transactions on Graphics, 9 (1990), pp. 66–104, https://doi.org/10.1145/77635.77639.

[15] H. Edelsbrunner and E. P. Mücke, Three-dimensional alpha shapes, ACM Transactions On Graphics (TOG), 13 (1994), pp. 43–72.

[16] Y. S. Elshakhs, K. M. Deliparaschos, T. Charalambous, G. Oliva, and A. Zoilas, A comprehensive survey on Delaunay triangulation: Applications, algorithms, and implementations over CPUs, GPUs, and FPGAs, IEEE Access, 12 (2024), pp. 12562–12585, https://doi.org/10.1109/ACCESS.2024.3354709.

[17] J. Gamble, H. Chintakunta, and H. Krim, Applied topology in static and dynamic sensor networks, 2012, https://doi.org/10.1109/SPCOM.2012.6290237.

[18] R. Ghassemi and S. Krishnan, Positive Alexander duality for pursuit and evasion, SIAM Journal on Applied Algebra and Geometry, 1 (2017), pp. 308–327, https://doi.org/10.1137/16M1089083.

[19] P. Hall, Introduction to the theory of coverage processes, Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York, 1988, https://doi.org/10.1002/0167-0115(88)90159-0.

[20] A. Hatcher, Algebraic Topology, Cambridge University Press, Cambridge, 2002.

[21] K. Igusa, Higher Franz-Reidemeister torsion, vol. 31 of AMS/IP Studies in Advanced Mathematics, American Mathematical Society, Providence, RI; International Press, Somerville, MA, 2002, https://doi.org/10.1016/s0550-3213(00)00739-3.

[22] A. M. Kellerer, On the number of clumps resulting from the overlap of randomly placed figures in a plane, J. Appl. Probab., 20 (1983), pp. 126–135, https://doi.org/10.2307/3213726.

[23] M. Kerber and H. Edelsbrunner, 3d kinetic alpha complexes and their implementation, in 2013 Proceedings of the Fifteenth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, 2013, pp. 70–77.

[24] D. Khryashchev, J. Chu, M. Vejdemo-Johansson, and P. Ji, A distributed approach to the evasion problem, Algorithms (Basel), 13 (2020), pp. Paper No. 149, 13, https://doi.org/10.3390/a13060149.

[25] H. La and W. Sheng, Flocking control of a mobile sensor network to track and observe a moving target, 2009, pp. 3129–3134, https://doi.org/10.1109/ROBOT.2009.5152747.

[26] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, Mobility improves coverage of sensor networks, 2005, pp. 300–308, https://doi.org/10.1109/1062689.1062728.

[27] B. Liu, O. Dousse, P. Nain, and D. Towsley, Dynamic coverage of mobile sensor networks, IEEE Transactions on Parallel and Distributed Systems, 24 (2013), pp. 301–311, https://doi.org/10.1109/TPDS.2012.141.

[28] B. Mohar and C. Thomassen, Graphs on surfaces, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2001.

[29] P. A. P. Moran, The volume occupied by normally distributed spheres, Acta Math., 133 (1974), pp. 273–286, https://doi.org/10.1007/BF02392147.

[30] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Transactions on Automatic Control, 51 (2006), pp. 401–420, https://doi.org/10.1109/TAC.2005.864190.

[31] R. Olfati-Saber, Distributed Kalman filtering for sensor networks, 2007, pp. 5492–5498, https://doi.org/10.1109/CDC.2007.4434303.

[32] R. Olfati-Saber, J. Fax, and R. Murray, Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE, 95 (2007), pp. 215–233, https://doi.org/10.1109/JPROC.2006.887293.

[33] R. Olfati-Saber and P. Jalalkamali, Coupled distributed estimation and control for mobile sensor networks, IEEE Transactions on Automatic Control, 57 (2012), pp. 2609–2614, https://doi.org/10.1109/TAC.2012.2190184.

[34] V. Pascucci, G. Scorziello, P.-T. Bremer, and A. Mascarenhas, Robust on-line computation of Reeb graphs: simplicity and speed, in ACM SIGGRAPH 2007 papers, 2007, pp. 58–es.

[35] Y. Qu, S. Xu, C. Song, Q. Ma, Y. Chu, and Y. Zou, Finite-time dynamic coverage for mobile sensor networks in unknown environments using neural networks, Journal of the Franklin Institute, 351 (2014), pp. 4838–4849, https://doi.org/10.1016/j.jfranklin.2014.05.011.

[36] G. Schroeder, Distribution of number of point targets killed and higher moments of coverage of area targets, Naval research logistics quarterly, 31 (1984), pp. 373–385, https://doi.org/10.1002/nav.3800310304.

[37] H. Su, X. Chen, M. Chen, and L. Wang, Distributed estimation and control for mobile sensor networks with coupling delays, ISA Transactions, 64 (2016), pp. 141–150, https://doi.org/10.1016/j.isatra.2016.04.025.
Algorithm 3: Single Adaptive Time-Step

**Input**: Sensor positions $X_n$, alpha complex $A_n$, the boundary cycles $B_n$, boundary cycle labelling $L_n: B_n \rightarrow \{0, 1\}$, current time $t$, timestep size $\Delta t$, sensing radius $R$

**Output**: Sensor positions $X_{n+1}$, alpha complex $A_{n+1}$, the boundary cycles $B_{n+1}$, boundary cycle labelling $L_{n+1}: B_{n+1} \rightarrow \{0, 1\}$

begin

$X_{n+\frac{1}{2}} \gets \text{UpdatePositions}(X_n, dt)$;
$A_{n+\frac{1}{2}} \gets \text{AlphaComplex}(X_{n+\frac{1}{2}}, R)$;
$B_{n+\frac{1}{2}} \gets \text{ComputeBoundaryCycles}(A_{n+\frac{1}{2}}, X_{n+\frac{1}{2}})$;
Remove boundary cycle corresponding to the outside of fence from $B_{n+\frac{1}{2}}$;

if Atomic Transition occurred then

$L_{n+1} \gets \text{LabelUpdate}(B_n, B_{n+\frac{1}{2}}, A_{n+\frac{1}{2}}, L_n)$ via Algorithm 2 or 5;
return $X_{n+\frac{1}{2}}, A_{n+\frac{1}{2}}, B_{n+\frac{1}{2}}, L_{n+1}$

else

$X_{n+1}, A_{n+1}, B_{n+1}, L_{n+1} \gets \text{AdaptiveTimeStep}(\frac{\Delta t}{2}, t, R, X_n, A_n, B_n, L_n)$;
$X_{n+\frac{1}{2}}, A_{n+\frac{1}{2}}, B_{n+\frac{1}{2}}, L_{n+\frac{1}{2}} \gets \text{AdaptiveTimeStep}(\frac{\Delta t}{2}, t + \frac{\Delta t}{2}, R, X_{n+\frac{1}{2}}, A_{n+\frac{1}{2}}, B_{n+\frac{1}{2}}, L_{n+\frac{1}{2}})$;
return $X_{n+1}, A_{n+1}, B_{n+1}, L_{n+1}$

end
Algorithm 4: Update Labelling

Input: Alpha complex $A_{n+1}$, the boundary cycles $B_n$ and $B_{n+1}$, boundary cycle labelling $L_n$: $B_n \rightarrow \{0, 1\}$, fence sensors $F$

Output: The mapping $L_{n+1}: B_{n+1} \rightarrow \{0, 1\}$, indicating whether each boundary cycle may contain an intruder

\begin{algorithm}
\input{alg_update_labeling}
\end{algorithm}
Algorithm 5: Update Labelling with Disconnected Components

**Input:** Alpha complex $A_{n+1}$, the boundary cycles $B_n$ and $B_{n+1}$, boundary cycle labelling $L_n: B_n \rightarrow \{0, 1\}$, fence sensors $F$

**Output:** The mapping $L_{n+1}: B_{n+1} \rightarrow \{0, 1\}$, indicating whether each boundary cycle may contain an intruder

begin
    for $b \in \{y \in B_{n+1} \cap B_n \mid \text{nodes of } y \text{ are connected to fence}\}$ do
        $L_{n+1}[b] = L_n[b]$;
    end
    switch Atomic transition do
        case 1-simplex added causing re-connection do
            $V = \{b \in B_n \setminus B_{n+1} \mid b \text{ is connected to } F\}$;
            for $b \in B_{n+1}$ do
                if $b$ is connected to $F$ but has no label in $L_n$ then
                    $L_{n+1}[b] \leftarrow \bigvee_{x \in V} L_n[x]$
                end
                else if $b$ is connected to $F$ then
                    $L_{n+1}[b] \leftarrow L_n[b]$
                end
            end
        end
        case 1-simplex removed causing disconnection do
            $V = \{b \in B_{n+1} \mid b \text{ is not connected to } F \text{ but has label in } L_n\}$;
            for $b \in B_{n+1} \setminus B_n$ do
                if $b$ is connected to $F$ then
                    $L_{n+1}[b] \leftarrow \bigvee_{x \in V} L_n[x] \lor L_n[b]$
                end
            end
        end
        otherwise do
            for $b \in B_{n+1} \setminus B_n$ do
                if $b$ is connected to $F$ then
                    $L_{n+1}[b] \leftarrow \bigvee_{x \in B_n \setminus B_{n+1}} L_n[x]$
                end
            end
        end
    end
    for $b \in B_{n+1}$ do
        if Nodes of $b$ form a 2-Simplex and are connected to $F$ then
            $L_{n+1}[b] \leftarrow \text{False}$
        end
    end
    return $L_{n+1}$;
end