Free-Space Ellipsoid Graphs for Multi-Agent Target Monitoring

Aaron Ray[1], Alyssa Pierson[2], Daniela Rus[1]

***Abstract*— We apply a novel framework for decomposing and reasoning about free space in an environment to a multi-agent persistent monitoring problem. Our decomposition method represents free space as a collection of ellipsoids associated with a weighted connectivity graph. The same ellipsoids used for reasoning about connectivity and distance during high level planning can be used as state constraints in a Model Predictive Control algorithm to enforce collision-free motion. This structure allows for streamlined implementation in distributed multi-agent tasks in 2D and 3D environments. We illustrate its effectiveness for a team of tracking agents tasked with monitoring a group of target agents. Our algorithm uses the ellipsoid decomposition as a primitive for the coordination, path planning, and control of the tracking agents. Simulations with four tracking agents monitoring fifteen dynamic targets in obstacle-rich environments demonstrate the performance of our algorithm.**

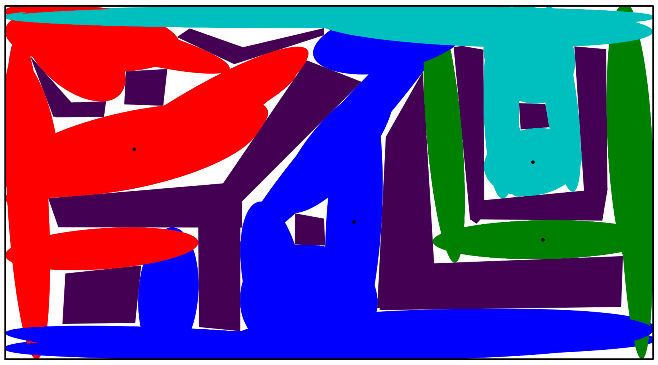


Fig. 1. Example ellipsoid decomposition of the “Jagged” environment and region assignment for team of tracking agents, with obstacles shown in purple and region assignments shown in red, green, blue, and cyan.

I. Introduction

As large-scale nonlinear programming solvers have become more performant, many applications have arisen where autonomous agents use an optimization-based control framework to simultaneously optimize the desired trajectory and necessary control inputs. Recent examples include Model Predictive Control (MPC) for vehicles entering and exiting a platoon [1], UAVs flying in formation subject to radio path loss constraints [2], and videography drones flying subject to visibility and aesthetic constraints [3]. However, collision avoidance constraints that prevent crashes with the surrounding environment can present problems when scaling to more general environments. The algorithms often rely on strict assumptions about obstacle structure, do not scale well with environment complexity, or are difficult to implement efficiently for real-time execution. Moreover, it can be difficult to connect the local collision avoidance constraints with global progress of the agent. It is desirable to reason about high level task coordination with the same framework used to eventually enforce collision-free motion.

We present an algorithm that utilizes an ellipsoidal decomposition of the environment to unify reasoning about high level planning and lower level optimization-based control. Our algorithm begins by decomposing the operational area into a set of ellipsoidal, obstacle-free regions. We use a novel method to connect these regions in a graph structure that reflects both connectivity and distance. Agent coordination can be reasoned about in the graph representation, and the same geometric primitives used to decompose the space are used as state constraints in an MPC algorithm. The algorithm scales very well with the complexity of the environment and is directly applicable to both 2D and 3D domains.

We demonstrate the decomposition framework in a multi-agent persistent patrolling task evocative of automated wild life census, behavior logging, event tracking, or specialized close-up videography. These applications require capabilities for tracking of specific features of the dynamic objects within given constraints. A key requirement of such capabilities is high-level coordination between agents to ensure each agent’s actions are useful to the whole. We demonstrate that our decomposition framework is useful in both the task coordination and low level control for a team of N tracking agents patrolling a group of M target agents. Our decomposition-based assignment algorithm performs favorably compared to obstacle-oblivious Voronoi-based assignment methods in two simulation environments.

The main contributions of the paper are:

- A novel method of decomposing and reasoning about free space in obstacle-rich environments that serves as a primitive for task assignment, path planning, and control
- An MPC implementation that leverages the free space decomposition generated by higher-level planning
- A demonstration of the proposed framework with a complex multi-agent persistent monitoring problem

The rest of the paper is organized as follows: We present related work in Section II. Section III presents our central decomposition and graph representation algorithm. We combine the elllipsoidal free-space representation with MPC in Section IV. Finally we demonstrate these techniques in a multi-agent monitoring problem in Section V.

II. Related Work

The fields of computer graphics, computational geometry, and robotics have extensive literature on polygonal decom-

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {aray, rus}@csail.mit.edu

[2] Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA pierson@bu.edu

This work was supported by the Office of Naval Research Grant N00014-18-1-2830, SUTD-DSO Project Astralis, and Boston University startup funds. We are grateful for their support.

positions. Seidel's trapezoidal algorithm [4] is a popular option for 2D decompositions. The Delaunay triangulation [5] is also often used in 2D and 3D. While both of these algorithms are efficient and provide complete decompositions of a region, the small individual pieces that they generate are a poor match for constraints in optimization-based control. We would instead prefer a smaller number of large regions in the decomposition. Finding the minimum number of convex polyhedra to cover a space is NP-Hard [6], [7], although there has been work in the direction of approximate minimal covers. Unfortunately existing solutions in this direction have various shortcomings (e.g. polynomial but large runtime [8] or restrictions to 2D [9]) that make them poor options for the 2D and 3D planning and control tasks that we have in mind. However, minimizing the total number of elements in the decomposition is less important than the quality of each. Even if the number of convex elements is not minimal, if each is large then a path between two points may only require traversing a small number of regions. To this end, we build upon the IRIS algorithm [10] that finds a single large convex region around a seed point.

The ideas in [11] are the closest to our own. In [11], the authors also decompose the free space into a collection of large convex regions to aide path planning. In contrast, our fixed-size representation of each convex region is more appropriate for optimization-based control, and our novel distance metric for building the connectivity graph is more general-purpose than their task-specific graph building.

Our example patrolling task draws inspiration from prior work in multi-agent target-tracking, coverage, persistent monitoring, and pursuer-evader games. One approach to tracking multiple targets with multiple agents is to divide the environment into regions among the trackers, such that each tracker is responsible for their subset of the environment. These coverage control approaches have been demonstrated in non-convex domains [12], [13], [14], [15] or when the total number of targets may be unknown [16], [17]. These coverage problems can also include constraints on viewpoints [18], or incorporate additional camera controls [19], [20] of the tracking vehicles. In persistent monitoring problems, the tracking agents must design strategies to continuously revisit targets within an environment, such as minimizing the time between observations of a given target [21]. Other work focuses on static targets [22] or non-convex environments [23].

Many of these existing works either simplify dynamic and kinematic constraints in order to provide capture guarantees, or consider the full reachability analysis at great computational cost. We demonstrate that our ellipsoid-based graph representation of the environment can address the target assignment and agent control problems in a unified framework and supports realistic, general dynamics models and realtime control. In the same vein as the author's prior work on cooperative target tracking with MPC [24], we show the benefits of jointly reasoning about high-level task coordination and low level control. Here, we focus on how our novel decomposition enables the high level task planning to be carried out with the same geometric primitives used to ensure safe motion by the MPC.

III. Ellipsoid Decomposition and Representation

A. Decomposition

We compute an approximate decomposition of the environment's obstacle-free space as a union of convex regions using the IRIS library [10]. Given a seed point, IRIS finds hyperplanes that separate the seed point from all obstacles. The seed point is expanded to the maximum volume ellipsoid enclosed by the hyperplanes, and the process repeats by finding new hyperplanes that separate the ellipsoid from the obstacles. We generate a lattice $\mathcal{L}$ of points with spacing d and begin with no regions in the free-space decomposition $\mathcal{E}$. Then, for each point in $l \in \mathcal{L}$, if it is not contained by any region in $\mathcal{E}$, we use IRIS to add a new obstacle-free region with l as the seed point, detailed in Algorithm 1.

The regions returned by IRIS can be represented in two ways: the bounding hyperplanes of the polyhedral regions, or the maximum volume ellipsoid constructed within them. Using the region defined by the hyperplanes is a more complete decomposition, but we prefer the ellipsoids for two reasons. First, the description of each region (a centroid and shape matrix) is of a constant size, while the polyhedral regions have varying numbers of sides. Constant-size region description yields a more desirable MPC implementation, as the resulting nonlinear program also has a constant size. Second, ellipsoid-based region descriptions lead to an elegant method of spatially relating free-space regions to each other. We denote ellipsoids as $\mathcal{M}(M, \mathbf{m}) = \{x \mid \|x - \mathbf{m}\|_M \leq 1\}$, where $\|w\|_A^2 = w^T A w$.

Algorithm 1 Ellipsoid Decomposition

```
1:  procedure EllipsoidDecomposition($\mathcal{O}$)
2:      $\mathcal{E} = \emptyset$                          ▷ Ellipsoid set initialized empty
3:      $\mathcal{L} \leftarrow$ LatticePoints($d$)        ▷ Generate lattice points
4:      $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L} \cap \mathcal{O}$        ▷ Cull points in obstacles
5:      while $|\mathcal{L}| > 0$ do
6:          $l \leftarrow l \in \mathcal{L}$             ▷ Choose new lattice point
7:          $\mathcal{L} \leftarrow \mathcal{L} \setminus l$            ▷ Remove point from set
8:          $\mathcal{E} \leftarrow$ InflateEllipsoid($l, \mathcal{O}$) $\cup\ \mathcal{E}$      ▷ Inflate
        ellipsoid from lattice seed point
9:          for $p \in \mathcal{L}$ do
10:             if $p \in \mathcal{E}$ then
11:                 $\mathcal{L} \leftarrow \mathcal{L} \setminus p$       ▷ Cull points covered by
        ellipsoid set
12:     return $\mathcal{E}$
```

B. Graph Construction

To facilitate using this collection of ellipsoids for spatial planning, we construct a graph $\mathcal{G} = (V, \mathcal{E})$ that represents the connectivity of the ellipsoids. Each ellipsoid $\mathcal{E}_j \in \mathcal{E}$ is represented by a vertex $v_j \in V$, and two vertices share an edge if their associated ellipsoids intersect. We next present an efficient method for checking ellipsoid intersection, which has the benefit of finding a point in the intersection of the ellipsoids if one exists.

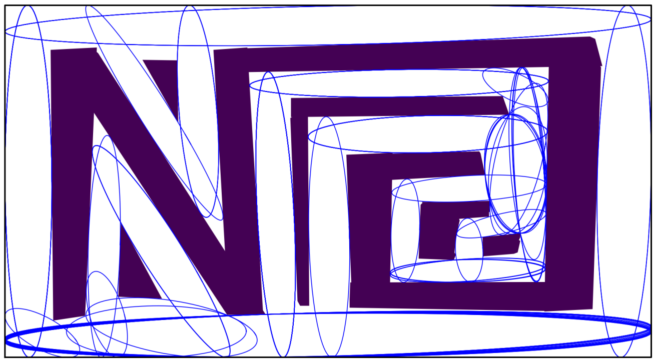


Fig. 2. Example ellipsoid decomposition generated by Algorithm 1. Obstacles are represented in purple and the free-space ellipsoids in blue.

For two ellipsoids $\mathcal{A}(A, a)$ and $\mathcal{B}(B, b)$, let

$$E_\lambda = \lambda A + (1-\lambda)B, \quad m_\lambda = E_\lambda^{-1}(\lambda A a + (1-\lambda)B b),$$

for some $\lambda \in [0,1]$. As proven in [25], m_λ is in both ellipses simultaneously for some value of λ on $[0,1]$ if and only if the two ellipsoids intersect. For $\lambda = 0$ and $\lambda = 1$, m_λ is equal to the origin of ellipse $\mathcal{B}$ and $\mathcal{A}$ respectively. A bisection search on λ will either return a point in the intersection of $\mathcal{A}$ and $\mathcal{B}$, or a point $m_{\tilde\lambda}$ that is not in either ellipsoid–a certificate that the ellipsoids do not intersect.

The correctness of our bisection search relies on m_λ moving monotonically from $\mathcal{B}$ to $\mathcal{A}$. Proposition 1 proves the monotonic behavior for axis-aligned ellipsoids with a diagonal shape matrix. In practice, we observe that monotonicity of m_λ holds for general ellipsoids.

Proposition 1. *Consider two axis-aligned ellipsoids $\mathcal{A}(A, a)$ and $\mathcal{B}(B, b)$ with diagonal shape matrices A and B. $\|m_\lambda - b\|_{\mathcal{B}}$ is monotonically increasing for $\lambda \in [0,1]$.*

Proof. First note that $\|m_\lambda - b\|$ is translation invariant. Without loss of generality, let $b = 0$. Making use of the fact that diagonal matrices commute with each other,

$$\begin{aligned}
&\|m_\lambda\|_{\mathcal{B}}^2 = \\
&\quad a^T A^T(\lambda A + (1-\lambda)B)^{-T}B(\lambda A + (1-\lambda)B)^{-1}Aa \\
&\quad = \sum_i a_i^2 B_{ii}\left(\frac{\lambda A_{ii}}{\lambda A_{ii} + (1-\lambda)B_{ii}}\right)^2 \\
&\quad = \sum_i a_i^2 B_{ii}\left(\frac{\lambda}{\lambda + (1-\lambda)\frac{B_{ii}}{A_{ii}}}\right)^2 .
\end{aligned}$$

As A and B are positive definite, $\frac{B_{ii}}{A_{ii}} \in (0, \infty)$. Additionally,

$$\frac{\lambda}{\lambda + (1-\lambda)\frac{B_{ii}}{A_{ii}}},$$

is monotonically increasing in $\lambda \geq 0$, for any $\frac{B_{ii}}{A_{ii}} \in (0, \infty)$. As $a_i^2 B_{ii}$ is also positive, $\|m_\lambda\|_{\mathcal{B}}^2$ is monotonically increasing for $\lambda \in [0,1]$. Squaring a positive function preserves monotonicity, so $\|m_\lambda - b\|_{\mathcal{B}}$ is monotonically increasing for $\lambda \in [0,1]$, thus completing our proof. □

The length of the path m_λ is an upper bound on the collision-free distance between the centroids of two ellipsoids. We define a distance heuristic $\hat{d}$ as

$$\hat{d}(\mathcal{E}_1, \mathcal{E}_2) = \int_0^1 \left\| \frac{dm_\lambda}{d\lambda} \right\| d\lambda,$$

which we compute numerically. The weighted edge set E of the ellipsoid connectivity graph is then defined by

$$E = \left\{ (i, j, \hat{d}(\mathcal{E}_i, \mathcal{E}_j)) \mid \mathcal{E}_i \cap \mathcal{E}_j \neq \emptyset \right\},$$

for $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}$, where $e = (i, j, w)$ denotes an edge from vertex i to vertex j with weight w. The resulting edge weights may violate the triangle inequality. To make the graph metric and provide a tighter bound on the shortest distance between ellipsoid centers, the weight of each edge can be updated to the weight of the lowest-weight path between its two endpoints in the original graph.

The graphical interpretation of environment free space, G, and the associated obstacle-free ellipsoids can now be used for task assignment and path planning, while the associated ellipsoids are useful for collision-free motion planning.

C. Path Planning

To move around the environment while avoiding obstacles, a mobile agent must plan a path in the ellipsoidal decomposition graph G from its current location x_{start} to some x_{goal}. To find such a path, we augment G with an additional vertex for the start and goal positions. Each of the two additional vertices shares an edge with vertices corresponding to the ellipsoids containing that point. The augmented vertex set is $V' = V \cup \{v_{start}, v_{goal}\}$, and the augmented edge set is

$$\begin{aligned}
E' = E &\cup \{(v_{start}, j, \|x_{start} - e_j\|) \mid \|x_{start} - e_j\|_{\mathcal{E}_j} \le 1\} \\
&\cup \{(v_{goal}, j, \|x_{goal} - e_j\|) \mid \|x_{goal} - e_j\|_{\mathcal{E}_j} \le 1\},
\end{aligned}$$

for $\mathcal{E}_j(E_j, e_j) \in \mathcal{E}$.

The shortest path from v_{start} to v_{goal} in the augmented graph $G' = (V', E')$, which we denote $\mathcal{E}^*$, corresponds to an upper bound on the minimum collision-free distance from x_{start} to x_{goal}. If the start and goal points are within the same ellipsoid, this pathfinding method may return a longer ellipsoid sequence than the single shared ellipsoid, depending on their position relative to the current ellipsoid's center. We explicitly check for this case, and if the start and end points share an ellipsoid we set $\mathcal{E}^*$ to be the current ellipsoid. In the patrolling case study, we pre-compute the shortest path between all pairs of nodes with the Floyd-Warshall algorithm [26, Chapter 5].

We use $\mathcal{E}^*$ to construct a sequence of waypoints between x_{start} and x_{goal}. A point in the intersection of each successive pairs of ellipsoids is added to the sequence, computed as described in Section III-B.

IV. Ellipsoid Decomposition with MPC

Given a sequence of waypoints from the graph-based path planning algorithm, we use a flexible Model Predictive Control (MPC) formulation to solve for the agent control inputs. Progress toward the next waypoint is achieved with a distance-based stage cost over the planning horizon, and a

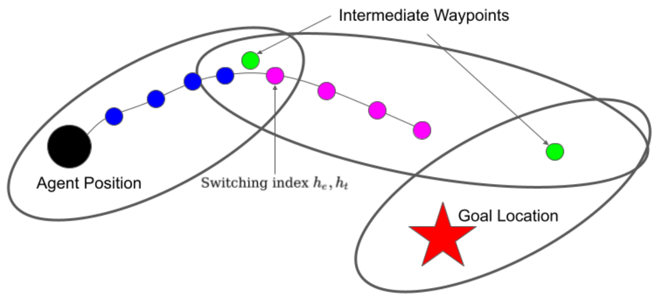


Fig. 3. Example of switching indices h_t and h_e. The first half of the trajectory (in blue) optimizes cost $l^1(x)$ based on distance to intermediate waypoint. The second half of the trajectory (in magenta) optimizes cost $l^2(x)$ toward the following waypoint. The switching indices are updated between MPC iterations depending on the predicted trajectory. In general, h_e and h_t need not be the same.

dynamically updated "cost switching" index ensures further progress toward the following waypoint. A similar updating index constrains each point on the planning horizon to be inside one of two free-space ellipsoids.

A. Cost Switching

To make progress toward the final goal position, we expect the agent to move toward the next waypoint in the sequence generated by the path planning algorithm. In practice, this intermediate waypoint may be within the planning horizon. It would be undesirable for the waypoint to be fixed for the entire planning horizon, which would lead to a trajectory that slows down or stops at the waypoint. Instead, part of the trajectory has a cost function related to the closest waypoint, while the rest is incentivized to reach the following waypoint.

We denote the cost functions related to the first and second waypoints as $l^1(\mathbf{x})$ and $l^2(\mathbf{x})$. In the simple case where the agent's only objective is moving between a fixed start and end point, then l^1 and l^2 is the distance between the target and corresponding waypoint. Our case study in Section V shows an example of a more complicated cost function. We define a *cost switching index* h_t, which determines when the MPC switches from the primary cost function l^1 to the secondary cost function l^2. Initially, h_t is set to $N + 1$, one larger than the MPC horizon. Each time a solution is returned from the MPC, the predicted trajectory for the agent is compared to the intermediate waypoint location. h_t is set to the first index where the tracking trajectory enters within some radius of the waypoint. After this point, it spends the rest of the horizon optimizing for the secondary goal. As the agent approaches the first waypoint, h_t decreases and more of the trajectory is pulled toward the second waypoint. This effect is illustrated in Figure 3.

B. Constraint Switching

Staying within the union of obstacle-free ellipsoids E allows an agent to avoid collisions with the environment. However, the union of ellipsoids is nonconvex, and imposing the constraint directly onto the MPC increases solution times and leads to poor local optima. Previous works [27], [28] address this issue by formulating the constraints as a mixed-integer convex nonlinear program, where exactly one convex region is the active constraint at each timestep. Unfortunately, requiring the solver to support integer constraints removes the possibility of using faster continuous nonlinear solvers. Instead, we employ a *constraint switching index* h_e analogous to the cost switching index that controls which free-space ellipsoid constrains each stage of the MPC.

The ellipsoids that we choose come from the path planning solution E^*. The first ellipsoid contains the agent and first waypoint. The second ellipsoid contains the first and second waypoints. We denote the first and second collision avoidance ellipsoids as $\mathcal{M}_1(M_1, \mathbf{m}_1)$ and $\mathcal{M}_2(M_2, \mathbf{m}_2)$. The first ellipsoid constraint is active until step h_e of the MPC horizon. The second ellipsoid constraint is active from step h_e onward. This results in a pair of quadratic constraints for each agent, $\|\mathbf{x}_k - \mathbf{m}_i\|^2_{\mathcal{M}_i} \le d^i_k,\ i \in \{0, 1\}$, defined for each step k of the MPC horizon.

Setting d^i_k to 1 constrains $\mathbf{x}$ to be within ellipsoid $\mathcal{M}_i$ at time k. Setting d^i_k to ∞ turns off the constraint at time k. We set d^0_k to ∞ if $k \ge h_e$, and 1 otherwise. Similarly, d^1_k is equal to ∞ if $k < h_e$, and 1 otherwise. Thus, in the interval $[0, h_e)$, only the $\mathcal{M}_1$ ellipsoid constraint must be satisfied, and vice-versa for $[h_e, N - 1]$. Within each of these intervals, the feasible positions for $\mathbf{x}$ are convex, which yields more reliable MPC performance in practice. We update h_e in a similar manner to h_t, based on what portion of the previous MPC solution was contained in the second ellipsoid.

C. Full Formulation

The full MPC formulation can be written as a constrained nonlinear optimization problem:

$$\underset{\mathbf{w}_{1:N}, \mathbf{u}_{1:N}}{\operatorname{argmin}} \sum_{k=1}^{N} l^1_k(\mathbf{x}_k)\mathbb{1}(k < h_t) + l^2_k(\mathbf{x}_k)\mathbb{1}(k \ge h_t) + q(\mathbf{u}_k) \tag{1a}$$

$$\text{s.t.}\quad \mathbf{w}_1 = \mathbf{w}(0), \tag{1b}$$

$$\dot{\mathbf{w}}(t_{c;n}) = f(\mathbf{w}(t_{c;n}), \mathbf{u}(t_{c;n})), \tag{1c}$$

$$\|\mathbf{x}_k - \mathbf{m}_i\|^2_{\mathcal{M}_i} \le d^i_k,\ i \in \{0, 1\} \tag{1d}$$

$$\mathbf{w} \in \mathcal{W}, \mathbf{u} \in \mathcal{U},\quad \forall k \in \{1, \ldots, N\}. \tag{1e}$$

In our implementation, the system dynamics $\dot{\mathbf{w}} = f(\mathbf{w}, \mathbf{u})$ are enforced with a third order collocation method (1c), as discussed in [29]. $t_{c;n}$ denotes the time corresponding to the n^{th} collocation point, and the resulting constraint can be represented directly in terms of $\mathbf{w}_{1:N}$ and $\mathbf{u}_{1:N}$. A control effort regularization function q is added to the cost function to encourage smoother control inputs. We let $q(\mathbf{u})$ be proportional to $\|\mathbf{u}\|^2$. The first control input $\mathbf{u}_1$ is applied to the system, and then the optimization re-solved in a receding horizon manner.

Note that the number of constraints necessary to enforce collision-free motion is constant and independent of the environment's complexity. While a polytope representation of free space results in a varying number of halfplane constraints, here we always have two constraints based two ellipsoids. This greatly aids MPC implementations where the solver is pre-compiled or reused between iterations.

V. Persistent Monitoring Case Study

We consider a multi-agent patrolling problem to demonstrate how our ellipsoidal decomposition algorithm provides a useful tool for jointly reasoning about planning and control. A team of tracking agents must periodically visit a group of moving target agents from a relative viewpoint. We demonstrate a task assignment algorithm that uses the decomposition graph G to reason about task assignments for the agents, and then uses the ellipsoids associated with paths in G to impose collision avoidance constraints in a viewpoint-aware MPC. We evaluate the proposed algorithm in a 2D simulation in two different environments that feature non-convex obstacles of varying shape and size, shown in Figure 1 (Jagged Environment) and Figure 2 (Spiral Environment).

A. Problem Formulation

Here, a team of N trackers seeks to observe M targets, with target-centric viewpoints specified as objectives. The team of tracking agents must be controlled to minimize the time between observations of each target. For each target $m \in \{1, ..., M\}$ we denote the desired viewing direction $\boldsymbol{\eta}$, viewpoint tolerance θ, and maximum distance r_{max}. For a tracking agent to successfully observe target m, it must enter a circular arc relative to m with medial axis $\boldsymbol{\eta}$, angle θ, and radius r_{max}.

Tracking agents move subject to second order unicycle dynamics, with the state consisting of position, heading angle, linear velocity, and angular velocity: $\mathbf{w} = [x;\ y;\ \theta;\ v;\ \omega]$, and the control input consisting of longitudinal and angular acceleration: $\mathbf{u} = [a;\ \alpha]$ The dynamics evolve according to $\dot{\mathbf{w}} = [v\cos(\theta);\ v\sin(\theta);\ \omega;\ a;\ \alpha]$.

The tracking agents are constrained to have a velocity in the range $[0, 3] m/s$, with acceleration constrained to be in $[-1, 1] m/s^2$. Angular velocity is constrained to $[-3, 3]$ rad/s, with angular acceleration constrained to $[-2, 2]$ rad/s^2. The acceleration constraints and second order dynamics make this a nontrivial control problem, even before considering the viewpoint cost function.

The target agents are given random obstacle-free destination points in the environment and move according to a first order unicycle model with a constant velocity of $0.3 m/s$ (angular velocity is controlled directly). They plan intermediate waypoints with the method outlined in Section III-C, and heading is driven with a Proportional Derivative (PD) feedback controller. By virtue of the waypoint choice, the target agents are usually in obstacle-free space, although they may occasionally drive through obstacles.

B. Task Assignment

Before task execution, the approximate ellipsoidal decomposition of free space G is constructed. At runtime, each tracking agent is assigned a subset of the targets it is responsible for observing. This subset updates according to changes in distribution of targets. Each tracking agent plans an order for visiting each of its assigned targets. The assignment order is used to generate a collision-free reference path for the tracker, which is followed by an MPC.

We examine two complementary methods for the assignment step. The first method dynamically updates each agent's region of responsibility online. Let $\hat{D}(\mathbf{x}_i, \mathbf{x}_j)$ return the shortest distance based on the approximation from G'. Each target is assigned to the closest tracker, as estimated by $\hat{D}$. The second method finds a static partition of the environment and assigns each tracker agent to a fixed region. We partition the set of free-space ellipsoids among the tracking agents with a greedy k-center approximation problem. For a metric graph $G = (V, E)$, the k-center problem is to find a subset of vertices $V' \in V$ such that the maximum distance between any vertex in G and a vertex in V' is minimized. The problem is NP-Complete, but there is a simple heuristic that guarantees at most twice the optimal distance: with V' initialized with a single random node, add the node in V that is furthest from any node in V' and repeat $k-1$ times [30]. Such a decomposition is illustrated in Figure 1. Each tracking agent is assigned responsibility for the targets currently in its designated partition. While the regions have some overlap, they generally ensure that the tracking agents are distributed around the environment. The two methods of dividing targets can be used interchangeably, depending on whether it is desirable to have each agent confined to a known region.

Once the targets have been divided among the tracking agents, a greedy heuristic h is used to guide the tracker's visitation order of its assigned targets. The visitation order heuristic is a weighted combination of each target's distance and the amount of time since it has been surveyed. Formally,

$$h^i = \hat{D}(\mathbf{x}_{tracker}, \mathbf{x}^i_{target}) - w_{staleness}(t_{now} - t^i_{seen}).$$

The heuristic balances opportunism in viewing targets that are convenient with an incentive to seek out targets that have not been seen in a while. The targets are pursued in ascending order of h_i, although as the ordering is updated at every timestep only the first two targets affect the control policy.

C. MPC Viewpoint Cost

We define a viewpoint cost function that encodes the desire to view the target from a specific angle and distance. Consider a target at position $\mathbf{x}_{target}$ with a desired viewing direction $\boldsymbol{\eta}$. In order to get the desired observation of the target, the tracking agent must enter into an arc with opening angle θ around $\boldsymbol{\eta}$, and within distance r. Let $\mathbf{x}$ denote the vector from $\mathbf{x}_{target}$ to $\mathbf{x}_{tracker}$. The viewpoint cost function l is composed of a term that penalizes the deviation of $\mathbf{x}$ from the direction of $\boldsymbol{\eta}$ and the length of $\mathbf{x}$ from the desired viewing distance. The viewpoint cost function is defined as

$$l(x) = \frac{\mathbf{x}^T}{\|\mathbf{x}\|}\eta + (\|\mathbf{x}\|^2 - r^2)^2. \tag{2}$$

The position and direction of interest of the target are predicted over the horizon of the MPC planning. As a result, $l(\mathbf{x})$ is implicitly a function of time from the time-dependence of $\mathbf{x}$ and η. We make this explicit by notating $l_i(\mathbf{x})$ as the cost function relative to the target at time i. Our implementation predicts the target's future motion assuming no control inputs are applied to the target over the planning horizon. When a tracker agent's current position relative to its current target enters the desired viewing cone, we consider the target to have been visited. When the tracking and target agent share an ellipsoid or are in adjacent ellipsoids, the cost function l is used as cost l^1 or l^2 in (1). In this case, the cost switching index h_t can be updated based on when the predicted trajectory enters the target's viewing cone.

D. Simulation

The viewpoint-optimizing cost function (2) for each agent is optimized subject to the agent dynamics by implementing (1) in Casadi [31] and solving with IPOPT [32]. The MPC considers a five second horizon, comprised of 40 equally-spaced steps. It takes an average of 50ms per MPC solution on an Intel Core i7-9750H CPU @ 2.60GHz. The MPC is given its previous solution as an initial guess. The computation for decomposition and graph construction is dominated by the ellipse inflation at each seed point. The ellipse inflation takes under 6 seconds in both test environments.

Still frames from our video illustrating the tracking task within Environment 2 are shown in Figure 6. We simulate both the static and dynamic ellipsoid region assignment algorithms. We compare these algorithms to a static and dynamic obstacle-oblivious assignment methods. The first comparison is to a static, obstacle-oblivious Voronoi partition of the space, with the same seed points used as the k-centers defining the ellipsoid partition. The second comparison is a dynamic Voronoi assignment algorithm, where targets are assigned to the closest tracker at every iteration without consideration of obstacles. All four algorithms use the same path planning and MPC implementations. Figure 4 presents the mean time between visits for targets, and Figure 5 presents the maximum time between visits for the targets over 51 runs. Each run randomizes the initial positions of the target agents and simulates 625 seconds of the tracking task. The static region assignments are slightly different between runs due to the stochastic nature of the k-centers approximation algorithm.

In both domains, the algorithms with dynamically allocated regions of responsibility outperform the static region assignments. This is unsurprising, as dynamic partitioning enables tracking agents to directly respond to shifting distributions of the target agents. In both environments, the dynamic ellipsoid decomposition method outperforms the dynamic Voronoi method, for both metrics of mean visit delay and maximum visit delay. The static ellipsoid decomposition slightly underperforms the static Voronoi method in the Spiral environment and outperforms it in the Jagged environment. The static ellipsoid decomposition compares most favorably when there are larger free-space regions thinly separated by obstacles, which is better reflected in the Jagged environment. Overall, these simulations demonstrate the efficacy of our ellipsoid decomposition and path planning to perform complex multi-agent tasks.

VI. Conclusions and Future Work

This paper has demonstrated a novel method for decomposing and reasoning about spatial structure in obstacle-rich 2D and 3D environments. The obstacle aware ellipsoid-based graph decomposition enables high-level coordination between agents, path planning, and collision-free motion planning in a unified representation. We have demonstrated the utility of this framework with a multi-agent persistent monitoring task where an MPC algorithm leverages the decomposition to monitor a team of target agents from desired viewpoints. We hope to extend the applications of this ellipsoid framework to other multi-agent tasks as a coordination and planning primitive.

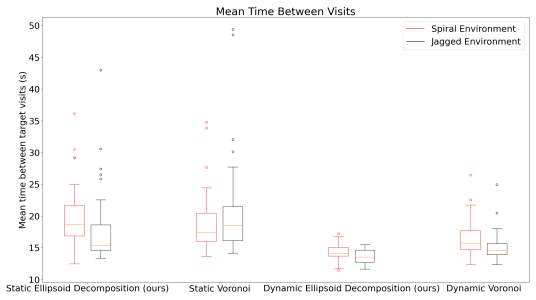


Fig. 4. Mean time between target visitation for the four algorithms.

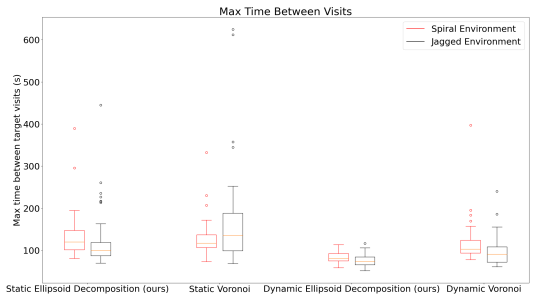
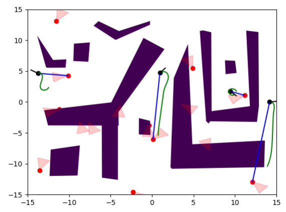
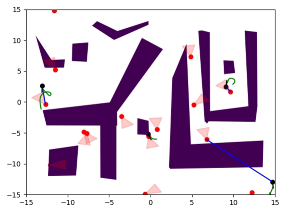


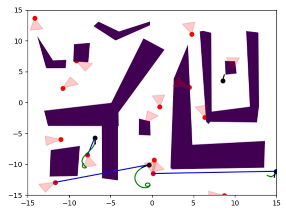
Fig. 5. Max time between target visitation for the four algorithms.



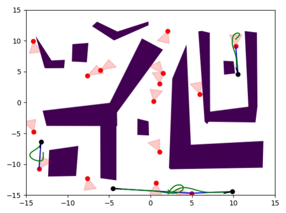
(a) $t = 0{:}25$s



(b) $t = 6{:}25$s



(c) $t = 18{:}75$s



(d) $t = 31{:}25$s

Fig. 6. Evolution of a tracking simulation over time. Four tracking agents are pictured as black dots. Purple blocks represent obstacles. Red dots represent targets of interest, and red triangles are relative viewpoint from which they must be viewed by a tracker. Green lines are the tracking agents' MPC plan at the current iteration. The dark blue line connects each tracker with its currently-assigned target.

References

[1] S. Graffione, C. Bersani, R. Sacile, and E. Zero, "Non-linear mpc for longitudinal and lateral control of vehicle's platoon with insert and exit manoeuvres," in *International Conference on Informatics in Control, Automation and Robotics*. Springer, 2020, pp. 497–518.

[2] A. Grancharova, E. I. Grøtli, D.-T. Ho, and T. A. Johansen, “Uavs trajectory planning by distributed mpc under radio communication path loss constraints,” *Journal of Intelligent & Robotic Systems*, vol. 79, no. 1, pp. 115–134, 2015.

[3] A. Ray, A. Pierson, H. Zhu, J. Alonso-Mora, and D. Rus, “Multi-robot task assignment for aerial tracking with viewpoint constraints,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1515–1522.

[4] R. Seidel, “A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons,” *Computational Geometry*, vol. 1, no. 1, pp. 51–64, 1991.

[5] D.-T. Lee and B. J. Schachter, “Two algorithms for constructing a delaunay triangulation,” *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[6] J. C. Culberson and R. A. Reckhow, “Covering polygons is hard,” *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pp. 601–611, 1988.

[7] M. Abrahamsen, “Covering polygons is even harder,” in *2021 Symposium on the Foundations of Computer Science*, 2021. [Online]. Available: https://arxiv.org/abs/2106.02335

[8] S. J. Eidenbenz and P. Widmayer, “An approximation algorithm for minimum convex cover with logarithmic performance guarantee,” *SIAM Journal on Computing*, vol. 32, no. 3, pp. 654–670, 2003. [Online]. Available: https://doi.org/10.1137/S0097539702405139

[9] H.-Y. Feng and T. Pavlidis, “Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition,” *IEEE Transactions on Computers*, vol. C-24, no. 6, pp. 636–650, 1975.

[10] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming,” in *Workshop on the Algorithmic Fundamentals of Robotics*, vol. 107, 2014.

[11] A. Sarmientoy, R. Murrieta-Cidz, and S. Hutchinsony, “A sample-based convex cover for rapidly finding an object in a 3-d environment,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3486–3491.

[12] Y. Stergiopoulos, M. Thanou, and A. Tzes, “Distributed collaborative coverage-control schemes for non-convex domains,” *IEEE Transactions on Automatic Control*, vol. 60, no. 9, pp. 2422–2427, 2015.

[13] S. Papatheodorou, A. Tzes, and Y. Stergiopoulos, “Collaborative visual area coverage,” *Robotics and Autonomous Systems*, vol. 92, pp. 126–138, 2017.

[14] A. Pierson and D. Rus, “Distributed target tracking in cluttered environments with guaranteed collision avoidance,” in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec 2017, pp. 83–89.

[15] L. Collins, P. Ghassemi, E. T. Esfahani, D. Doermann, K. Dantu, and S. Chowdhury, “Scalable coverage path planning of multi-robot teams for monitoring non-convex areas,” *arXiv preprint arXiv:2103.14709*, 2021.

[16] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, “Resilient active target tracking with multiple robots,” *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 2019.

[17] P. M. Dames, “Distributed multi-target search and tracking using the phd filter,” *Autonomous robots*, vol. 44, no. 3, pp. 673–689, 2020.

[18] M. Petrlík, V. Vonásek, and M. Saska, “Coverage optimization in the cooperative surveillance task using multiple micro aerial vehicles,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 4373–4380.

[19] W. Hönig and N. Ayanian, “Dynamic multi-target coverage with robotic cameras,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1871–1878.

[20] O. Arslan, H. Min, and D. E. Koditschek, “Voronoi-based coverage control of pan/tilt/zoom camera networks,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5062–5069.

[21] S. Alamdari, E. Fata, and S. L. Smith, “Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014. [Online]. Available: https://doi.org/10.1177/0278364913504011

[22] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to persistent monitoring tasks,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[23] J. M. Palacios-Gasós, D. Tardioli, E. Montijano, and C. Sagüés, “Equitable persistent coverage of non-convex environments with graph-based planning,” *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1674–1694, 2019. [Online]. Available: https://doi.org/10.1177/0278364919882082

[24] A. Pierson, A. Ataei, I. C. Paschalidis, and M. Schwager, “Cooperative multi-quadrotor pursuit of an evader in an environment with no-fly zones,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 320–326.

[25] I. Gilitschenski and U. D. Hanebeck, “A direct method for checking overlap of two hyperellipsoids,” in *2014 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2014, pp. 1–6.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[27] R. Deits and R. Tedrake, “Efficient mixed-integer planning for uavs in cluttered environments,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 42–49.

[28] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, “Aggressive quadrotor flight through cluttered environments using mixed integer programming,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1469–1475.

[29] R. Tedrake, “Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832),” chapter 10. [Online]. Available: http://underactuated.mit.edu/

[30] D. S. Hochbaum and D. B. Shmoys, “A best possible heuristic for the k-center problem,” *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985. [Online]. Available: http://www.jstor.org/stable/3689371

[31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[32] A. Wächter and L. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, pp. 25–57, 03 2006.