2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)
University of Southern California, LA, USA, December 4-5, 2017

Distributed Target Tracking in Cluttered Environments with Guaranteed Collision Avoidance

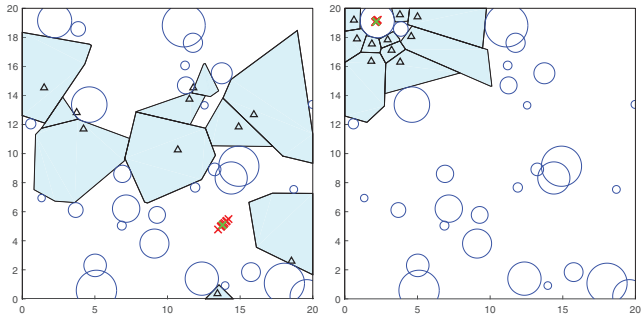
Alyssa Pierson and Daniela Rus

***Abstract*— We propose a distributed, online algorithm for a group of pursuer agents to track a target through a cluttered environment. The pursuer agents must avoid collision with the obstacles at all times. We introduce the Obstacle-Aware Voronoi Cell (OAVC), a modified Voronoi tessellation that dynamically weights the boundaries between agents and obstacles such that an agent's OAVC is tangent but never intersecting the obstacle. The agents plan their control actions within their OAVC, guaranteeing collision avoidance among themselves and other agents. We demonstrate that by using tools from Voronoi-based coverage control, the pursuers successfully track a target given only an estimate of its position. Simulations conducted in Matlab demonstrate the performance of our algorithm.**

I. Introduction

In this paper, we propose an algorithm for a group of agents to track a target through a cluttered environment while simultaneously avoiding obstacles. To guarantee collision avoidance, the agents calculate a safe area within the environment, and plan all actions within that safe area. We introduce the Obstacle-Aware Voronoi Cell (OAVC), a modified Voronoi tessellation that generates a safe, collision-free area. Using a position estimate of the target to generate a probability density across the environment, the agents calculate and move towards the centroid of their OAVC, allowing the agents to move closer to the predicted location of the target while avoiding collisions with obstacles and other agents. This algorithm builds upon the author's previous work, wherein a group of agents cooperatively tracked an evader through an environment over time [1].

Our algorithm is pertinent to a handful of emerging applications, such as search and rescue, security and surveillance, and robotic videography. Consider, for example, the target is a suspected criminal, and the agents are surveillance drones. The agents must track the suspect while avoiding buildings, bridges, trees, and other environmental obstacles. If the target enters a cluttered region, our algorithm allows the agents to pursue the target while avoiding the obstacles. Another application is tracking a lost person or animal in a forested area. In a previous work [1], we considered the problem of distributing the pursuers around a no-fly zone boundary while tracking an evader. In that paper, we utilized tools from Robust Model Predictive Control (MPC) to ensure collision-free paths for the quadrotor pursuers, and tools from Voronoi-based coverage control for distributing the agents about the no-fly zone boundary. When the evader was in free space, we assumed the pursuers had perfect knowledge of the targeted evader. For this work, the agents only have an estimate of the target's position, even in free space. This allows us to unify the control strategy to a single policy tracking the target in free space and inside an obstacle.



(a) (b)

Fig. 1: Simulation of target tracking for $n = 10$ pursuers (triangles) in an environment with $m = 30$ circular obstacles. The shaded regions are the OAVC of each agent, and the target estimate is given by the red X's, with the true position denoted with a green X.

While there has been extensive research on path planning and obstacle avoidance for mobile robots [2], [3], most algorithms rely on simple approximations of the robot dynamics [4], [5], [6], [7] . If the underlying robot is highly nonlinear, or subject to uncertainties, these approximations may result in controller instability and collision with obstacles. In [8], [1], a robust MPC using linear matrix inequalities (LMIs) demonstrated robust performance under modeling uncertainties and measurement noise. Combined with a path planning algorithm, collision avoidance in the presence of uncertainties was guaranteed. Here, we incorporate the collision avoidance into path planning by restricting the area the agent uses to choose its path.

To generate this safe-area for each agent, we draw inspiration from Voronoi-based coverage control. A Voronoi-based coverage strategy, first proposed by Cortés et al. [9], [10], drives all robots towards the centroids of their Voronoi cells. Also known as the move-to-centroid controller, this strategy can be extended beyond coverage control to other multi-agent problems. One application is tracking intruders within

This work was supported by the Office of Naval Research (ONR) Grant N00014-12-1-1000 and Toyota Research Institute (TRI). Toyota Research Institute provided funds to assist the authors with their research but this paper solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We are grateful for this support.

The authors are with the Computer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA [apierson, rus]@mit.edu

an environment [11], [12], however, these do not consider obstacles in the environment and cannot guarantee collision avoidance. Other works have used a Voronoi-based strategy for multiple cooperative pursuers against both a single evader [13], [14], [15], [16] and multiple evaders [17], but again do not account for obstacle avoidance.

Our algorithm utilizes a modified Voronoi cell to guarantee collision avoidance between agents and obstacles. One approach is to create a buffer zone within the Voronoi cells, such that any robot located on the boundary of its cell cannot collide with its neighbors [18], [19]. Here, these modified Voronoi cells can be used as safety regions for agents planning actions. The buffered cells in [18] and [19] use a static weighting to create the buffered zone in each cell - useful when the offset between agents is known and there are no obstacles. In contrast, our algorithm uses a dynamic weight between an agent and obstacles such that the boundary of the cell is tangent to the obstacle.

In this work, the agents use a modified Voronoi partition to divide the environment up into convex, safe regions. This partition is designed to dynamically modify the boundary calculation based on the obstacle positions, as well as divide the region among multiple agents. An estimate of the target's position is given to the agents, which is used to define a configuration cost of the agents. By choosing a controller that decreases the configuration cost, the agents are able to successfully track a target as it moves throughout the environment. The remainder of this paper is organized as follows. In Section II, we present our problem formulation. Section III presents our tracking strategy. We demonstrate the performance with Matlab simulations in Section IV, and present our conclusions in Section V.

II. Problem Formulation

Consider a bounded, convex environment $Q \subset \mathbb{R}^2$, with points in Q denoted q. For n agents, we denote the positions $p_i \in Q$, for $i \in \{1, ..., n\}$. We assume all agents have integrator dynamics,

$$\dot{p}_i = u_i, \tag{1}$$

where u_i is the desired control input for agent i. The agents track some target, T, through the environment. We denote the position of the target as p_T. The agents may not know the exact location of the target, and instead use an estimated position of the target. We define the position estimate with a probability density function $\phi_T(q)$, where the value of $\phi_T(q)$ represents the probability of target T being at location q. While $\phi_T(q)$ may take many forms, we assume $\phi_T(q) > 0$ and that over the environment, $\int_Q \phi_T(q)dq = 1$.

Each agent must also avoid obstacles in the environment. The agents may not know the location of all obstacles in the envrionment, but we assume that nearby obstacles can be detected. We represent the m static obstacles in the environment as circles, with each obstacle centered at position z_j with radius R_j. We refer to the entire set of obstacles as $Z = [..., z_j^T, ...]$. Obstacles are allowed to have varying radii, and may overlap, allowing for the creation of larger, non-convex obstacles from a series of overlapping circular obstacles. For each obstacle, let $\mathcal{Z}_{A_j}$ be the area it occupies in the environment, defined

$$\mathcal{Z}_{A_j} = \{q \in Q \mid \|q - z_j\| \leq R_j\}.$$

We also define $\mathcal{Z}_A$ as the total area occupied by all obstacles, calculated as $\mathcal{Z}_A = \bigcup_{j=1}^{m} \mathcal{Z}_{A_j}$.

A. Distributed Tracking

In this problem, we want the pursuers to track a target in a distributed fashion while avoiding collisions between obstacles and other agents. To guarantee collision avoidance, consider some convex safe area A_i for each pursuer. Given this safe area A_i, the pursuer must choose some control strategy u_i to remain inside A_i and move towards the estimated position of the target. First, we will introduce a strategy for a pursuer to choose u_i given an area A_i. Later, we present our formulation of A_i using a modified Voronoi cell and show its relation to Voronoi-based coverage control.

For each agent, we introduce a configuration cost $\mathcal{H}_i$ to assess its position tracking a target within its area A_i. For an agent at location p_i tracking a target with probability density $\phi_T(q)$, we define

$$\mathcal{H}_i = \int_{A_i} \|p_i - q\|^2 \phi_T(q) dq.$$

Intuitively, if the pursuer doesn't know the exact location of the target, they should move towards the most likely location of the target. The configuration cost $\mathcal{H}_i$ is low if the pursuer is near high values of $\phi_T(q)$, and the cost increases when the pursuer is further away from the target. If the area A_i is static and $\phi_T(q)$ does not evolve over time, one control policy to decrease the agent's cost $\mathcal{H}_i$ is to move towards the centroid. Analogous to a physical mass and centroid, we define

$$M_{A_i} = \int_{A_i} \phi_T(q) dq, \text{ and } C_{A_i} = \frac{1}{M_{A_i}} \int_{A_i} q \phi_T(q) dq. \tag{2}$$

It can be shown that a local minimum of the configuration cost $\mathcal{H}_i$ occurs when the agent is located at the centroid of its safe area A_i. To move the agents toward their centroid, we propose the following controller:

$$\dot{p}_i = u_i = k_i \left(C_{A_i} - p_i\right), \tag{3}$$

where $k_i > 0$ is a proportional gain and C_{A_i} is the centroid of the OAVC. Proposition 1 demonstrates that by moving towards its centroid, a pursuer can decrease its local configuration cost and converge upon a static location.

Proposition 1: Using the control strategy in (3), the pursuers converge to the centroid of their safe area,

$$\|p_i - C_{A_i}\| \to 0. \tag{4}$$

Proof: To prove (4), we invoke LaSalle's Invariance Principle [20]. First, we introduce a continuously-differentiable Lyapunov-like function $\mathcal{V}_i$ that takes the form of the coverage cost function $\mathcal{H}_i$. We show that all trajectories of the system are bounded, and that the function is

non-increasing, thus $\dot{\mathcal{V}}_i \leq 0$. We then use LaSalle's to prove the claims of the proposition. Consider the function

$$\mathcal{V}_i = \int_{A_i} \|p_i - q\|^2 \phi_T(q) dq,$$

with derivative

$$\dot{\mathcal{V}}_i = \frac{\partial \mathcal{V}_i}{\partial p_i} \dot{p}_i.$$

Using tools from locational optimization [21], it can be shown that

$$\frac{\partial V_i}{\partial p_i} = -M_{A_i} \left(C_{A_i} - p_i \right).$$

Substituting our controller (3) into $\dot{\mathcal{V}}_i$ yields

$$\begin{aligned} \dot{\mathcal{V}}_i &= -M_{A_i} \left(C_{A_i} - p_i \right)^T k_i \left(C_{A_i} - p_i \right), \\ &= -k_i M_{A_i} \| C_{A_i} - p_i \|^2 \leq 0. \end{aligned}$$

Given that the derivative $\dot{\mathcal{V}}_i \leq 0$, we see the trajectories for robots $p_i(t)$ are bounded. To complete the proof, we find the largest invariant set within the set defined by $\dot{\mathcal{V}}_i = 0$. By inspection, this occurs when $p_i = C_{A_i}$, and from our control law (3), this itself is an invariant set. Therefore, by LaSalle's Invariance Principle, we have that

$$p_i(t) \to C_{A_i} \text{ as } t \to \infty,$$

proving (4) from Proposition 1. ■

Proposition 1 provides a basic strategy for a single agent given a static, convex safe area A_i and probability density for the target $\phi_T(q)$. Under these assumptions, a good strategy is to move towards the centroid of its safe-area, guaranteeing it does not collide with any other obstacles or agents in the environment. In the following sections, we propose using tools from Voronoi-based coverage control to generate the safe area. First, we summarize key results from locational optimization, then present our modified Voronoi cell to use in environments with obstacles.

B. Locational Optimization

In the previous section, we presented a general formulation for a pursuer seeking a target given a pre-assigned safe region and probability density function for the target. For a convex safe area A_i, by moving to the centroid C_{A_i}, the pursuer decreases its individual configuration cost while avoiding collisions. We propose using a modified Voronoi cell as the safe area for each agent. In the absence of obstacles, our modified Voronoi cell is equivalent to the standard Voronoi partition. Using a Voronoi tessellation, we can consider a combined configuration cost of the group. Here, we summarize results from locational optimization stating the agents reach a local minimum of the combined configuration cost using a move-to-centroid control law. Consider the standard Voronoi partition, defined

$$V_i = \{ q \in Q | \ \|q - p_i\|^2 \leq \|q - p_j\|^2, \forall j \neq i, \quad i, j \leq n \}.$$

In the absence of obstacles, we see that if agents start within their cell and only plan actions inside V_i, they avoid collisions with all other agents. We write the configuration cost for the group of agents

$$\mathcal{H}(p_1, ...p_n) = \int_Q \min_{i \in \{1,...,n\}} \|q - p_i\|^2 \phi_T(q) dq. \quad (5)$$

Given the properties of the Voronoi partition, the configuration cost function becomes

$$\mathcal{H} = \sum_{i=1}^{n} \int_{V_i} \|p_i - q\|^2 \phi_T(q) dq.$$

We also define the mass M_{V_i} and centroid C_{V_i} of the Voronoi cell as calculated in (2). Despite the complex dependency between the position of the robots and the geometry of the Voronoi cells evolving over time, a surprising result from locational optimization [21] is that

$$\frac{\partial \mathcal{H}}{\partial p_i} = -\int_{V_i} \left(q - p_i \right) \phi_T(q) dq = -M_{V_i} \left(C_{V_i} - p_i \right),$$

which implies the critical points of $\mathcal{H}$ correspond to the robots positioned at the centroids of their Voronoi cells, or $p_i = C_{V_i}$ for all i. Critical points can correspond to either local minima, local maxima, or saddle points. Cortés introduced a controller that drives robots only to the local minima critical points of the cost function [9],

$$u_i = k_p \left(C_{V_i} - p_i \right). \quad (6)$$

In an environment with no obstacles, the move-to-centroid controller provides a good strategy to move towards a target with a known probability density function. When the density function is time-varying, it is possible to maintain coverage [11], [12], [22] with modifications to the controller and restrictions on the properties of $\phi(\cdot)$. Using these results, we propose a framework that uses a move-to-centroid controller with modified Voronoi cells that incorporate obstacles in the environment.

C. Obstacle-Aware Voronoi Cells

For environments with obstacles, one approach may be to treat obstacles as static agents in the environment. However, the standard Voronoi partition may intersect obstacles near the agent. Calculating a Buffered Voronoi Cell [19] creates a conservative bound on the safe area, but does not guarantee collision avoidance with larger obstacles. While collision avoidance could be incorporated on a secondary controller for the agent, it is desirable to have a unified control law. We propose a modified Voronoi cell formulation that differentiates between obstacles and other agents within the environment. When an agent computes its Voronoi boundary between itself and another agent, it calculates the boundary using the standard partition. For obstacles, the agent dynamically assigns weights such that the boundary of their modified Voronoi cell is tangent to the obstacle boundary. We call this an "Obstacle-Aware Voronoi Cell" (OAVC), defined

$$\begin{aligned} A_i = \{ q \in Q \mid \|p_i - q\|^2 \leq \|z_j - q\|^2 - w_{ij}, j \in Z \text{ and} \\ \|p_i - q\|^2 \leq \|p_k - q\|^2, k \in \{1, ..., n\} \neq i \} \end{aligned} \quad (7)$$

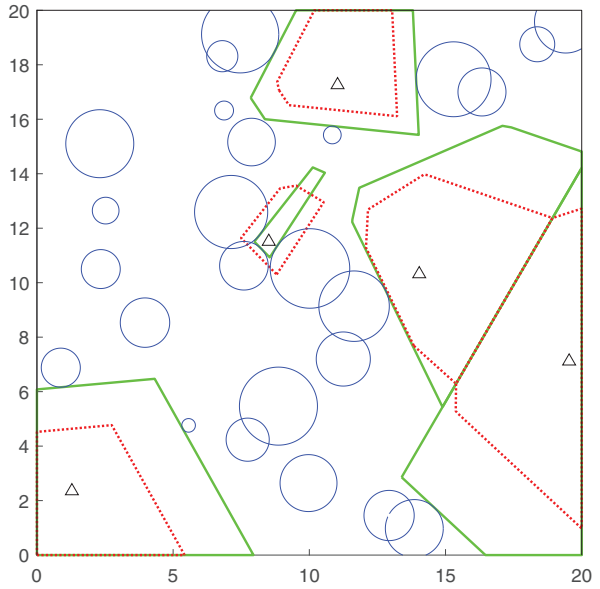


Fig. 2: Comparison of our Obstacle-Aware Voronoi cells (green) with regular Voronoi cells (red dash) around obstacles (blue circles). Our OAVC provides a larger safe region, but does not intersect obstacles at close proximity.

where w_{ij} is a dynamic weight that places the boundary line tangent to the obstacle boundary, defined

$$w_{ij} = 2R_j\|p_i - z_j\| - \|p_i - z_j\|^2.$$

Note that as the agents move, the value of w_{ij} changes, but the boundary line will always be tangent to the obstacle. Furthermore, agents only need to know the locations and radii of nearby obstacles to calculate their OAVC. Similar to the dynamic sensing radius proofs in [9], it can be shown that obstacles beyond a certain distance of the furthest vertex will not influence the boundary of the cell. From this, we assume that agents can sense the obstacles and other agents needed to generate their OAVC, but do not necessarily need the full position information of all obstacles and agents.

The purpose of the dynamic weight in the OAVC is to create the largest possible convex cell for the agent around static obstacles. By maintaining a convex cell, the agent can utilize a move-to-centroid controller and guarantee it will not collide with other agents or obstacles. Figure 2 illustrates a comparison of the standard Voronoi partition (red dash) versus our modified partition (green). In the figure, note that when an agent (triangle) is far away from an obstacle (blue circle), the Voronoi cell does not utilize all available free space for the agent. When an agent is near an obstacle, the standard Voronoi cell may intersect an obstacle. In contrast, our OAVC provides a larger area for agents far away from obstacles, yet ensures the boundaries never intersect nearby obstacles. Between two agents, both the regular Voronoi cell and our OAVC have the same edge. In the absence of obstacles, our partition in (7) reduces to the standard Voronoi partition among agents.

III. Tracking Strategy

The previous section summarized our problem formulation, and presented some basic results on locational optimization. As stated, in the absence of obstacles, the move-to-centroid controller would converge to a locally-optimal configuration around the target. In this section, we first present our tracking strategy given a position estimate of the target. Next, we provide an example of how a Kalman filter with a near-constant velocity target model can be used to generate the position estimate. Simulations demonstrating our overall algorithm are presented in Section IV.

The goal of the agents is to avoid obstacles while tracking the target. To guarantee collision avoidance while tracking, we propose the agents employ a move-to-centroid controller using the OAVC tessellation. Proposition 2 demonstrates under this control strategy, the agents will not collide with obstacles and other agents.

Proposition 2: For n agents at positions p_i, with non-overlapping starting locations $p_i(t_0) \notin \mathcal{Z}_A$ and OAVC A_i (7), the controller

$$u_i = k_p(C_{A_i} - p_i) \tag{8}$$

guarantees $p_i(t) \neq p_j(t)$, $\forall i \neq j$ and $p_i(t) \notin \mathcal{Z}_A$, $\forall, t > t_0$.

Proof: By the properties of A_i, each agent's cell is convex. It is known that the centroid of a convex shape lies inside the convex hull of the shape's vertices. Thus, at every time step, the agent will only move within its cell, avoiding collisions with obstacles and other agents. For agents with initial positions $p_i(t_0) \notin \mathcal{Z}_A$, we know that $A_i(t_0)$ does not intersect any obstacles or other agents. Since the agents will only move within their collision-free safe area, each subsequent calculation of $A_i(t)$ will never intersect obstacles Z. Thus, using the controller (8) with initial configurations $p_i(t_0) \notin \mathcal{Z}_A$ guarantees collision avoidance over time. ■

By Proposition 2, we know the agents avoid collisions with obstacles and other agents. However, we do not guarantee anything about the pursuers "capturing" a target. To present guarantees on capture, further structure is required on both the environment and the target dynamics. Instead, we present a general framework for agents to track static or dynamics targets while avoiding obstacles. For static targets, our approach provides coverage among obstacles, and by using the OAVC formulation, increases the covered area of each agent relative to the Voronoi tessellation. In our simulations, we also demonstrate that for a moving target, the agents follow the target through the environment while avoiding obstacles and the other agents, dynamically adjusting their coverage configuration. The main steps of our algorithm are summarized in Algorithm 1. In Algorithm 1, we do not address how the agents calculate the probability density function of the target, $\phi_T(q)$. The following sections provides one example of determining $\phi_T(q)$ using a Kalman filter.

A. Target Estimation with a Kalman Filter

Our tracking strategy proposes that the pursuers only need an estimate of the target's position to determine its probability density function $\phi_T(q)$ and calculate the pursuer

Algorithm 1 Distributed Tracking in Clutter

1: Calculate Obstacle-Aware Voronoi Cell A_i
2: Estimate Target Position and Distribution $\phi_T(q)$
3: Calculate Centroid C_{A_i}
4: Move to Centroid $\dot{p}_i = k_p(C_{A_i} - p_i)$

control law. While the target's position can be estimated in many ways, this section highlights one method of calculating $\phi_T(q)$. Here, the agents estimate the target's position using a Kalman filter and a near-constant velocity model of the target's movement. We use this example to illustrate how the pursuers might estimate a target location from observations and use that in their control law. This model is also used for the simulations presented in Section IV.

We propose using the Kalman filter to estimate the target's location to generate $\phi_T(q)$. Each agent may have its own estimate of the target, or the agents may share an estimate of the target. We assume the dynamics of the target are linear and Gaussian, the measurements are linear and Gaussian, and the initial distribution of the target's position is also Gaussian. We assume a normal distribution for the target's position, and update each agent's estimate of the target's mean and covariance matrix. This is consistent with Voronoi-based coverage control literature, wherein the "information density function" $\phi_T(q)$ is often generated as a Gaussian peak [9], [11], [23]. For the target with position p_T and velocity $\dot{p}_T$, define $x_k = [p_T \ \dot{p}_T]^T$ as the true position and velocity of the target at time k. We define $\hat{x}_k$ as the estimate of the mean, and P_k as the estimate of the covariance. The Kalman filter comprises two steps: a prediction step and an update step. We write the prediction step as [24]

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1},$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k,$$

where F_k is the state-transition model, and Q_k is the covariance of the process noise. For a near-constant velocity model of the target's movement,

$$F_k = \begin{bmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ Q_k = \tilde{q} \begin{bmatrix} \frac{t^4}{4} & 0 & \frac{t^3}{2} & 0 \\ 0 & \frac{t^4}{4} & 0 & \frac{t^3}{2} \\ \frac{t^3}{2} & 0 & t^2 & 0 \\ 0 & \frac{t^3}{2} & 0 & t^2 \end{bmatrix}.$$

We also define z_k as the measurement model for taking observations. We write

$$z_k = H_k x_k + v_k,$$

where $v_k \sim \mathcal{N}(0, R)$ and $H = [I \ 0 \mid 0 \ 0]$. The estimate of the covariance evolves as

$$y_k = z_k - H_k \hat{x}_{k|k-1},$$
$$S_k = H_k P_{k|k-1} H_k^T + R_k,$$
$$K_k = P_{k|k-1} H_k^T S_k^{-1},$$

where R_k is noise in observation and K_k is the optimal Kalman gain. Overall, the agents update the estimate of the mean and the covariance as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k,$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}.$$

By using the Kalman filter to determine an estimate of the mean and covariance, the agents then generate the $\phi_T(q)$ function to determine the centroid of their Obstacle-Aware Voronoi cell. Algorithm 2 summarizes this procedure, which is also used in the simulations.

Algorithm 2 Distributed Tracking with Kalman Filter

1: Calculate Obstacle-Aware Voronoi Cell A_i
2: Predict Target's Next Position
3: Make Observation of Target's Movement
4: Update Estimate of Target Mean and Covariance
5: Calculate $\phi_T(q)$
6: Calculate Centroid C_{A_i}
7: Move to Centroid $\dot{p}_i = k_p(C_{A_i} - p_i)$

IV. Simulations

Simulations run in Matlab demonstrate the performance of our algorithm. We present two simulation results: first, an example of a group of agents tracking a moving target. Next, we include an example where the tracking is limited by obstacles barricading the environment. Videos of our simulations are included with the submission of this paper. In both cases, the obstacles are initialized with random positions and radii, and obstacles are allowed to overlap. The agents and target were also assigned random starting positions. The target moves with a potential field repulsion from the other agents, with its velocity capped at a fixed speed. To initialize the Kalman filter for each agent, the robots predict the mean of the distribution is located in the center of the environment. Over time, as the robots update their estimate, this converges to the true location of the target.

A. Moving-Target Tracking

This scenario demonstrates the performance of the agents tracking a moving target. Here, $n = 10$ agents track a target in an environment with $m = 30$ obstacles. Figure 3 shows the configuration of all agents and obstacles over time. From this figure, we see the robots swarm around the target while simultaneously avoiding all obstacles in the environment.

Figure 4 illustrates the trajectories of the agents and the target over time. Here, we see that the target, shown as the red X, is allowed to move freely through the environment, passing through obstacles at times. We also see that the trajectories of the agents, illustrated as the black triangles, avoid all obstacles while following the target.

To analyze the performance of our agents, we examine the configuration cost (5) over time, shown in Figure 5. Initially, the agents are not near the target, resulting in a high initial configuration cost. As the agents track the target over time, the cost decreases. Since the target can increase its distance from the agents by entering obstacles, this cost increases slightly at times. Despite minor fluctuations in the cost, the

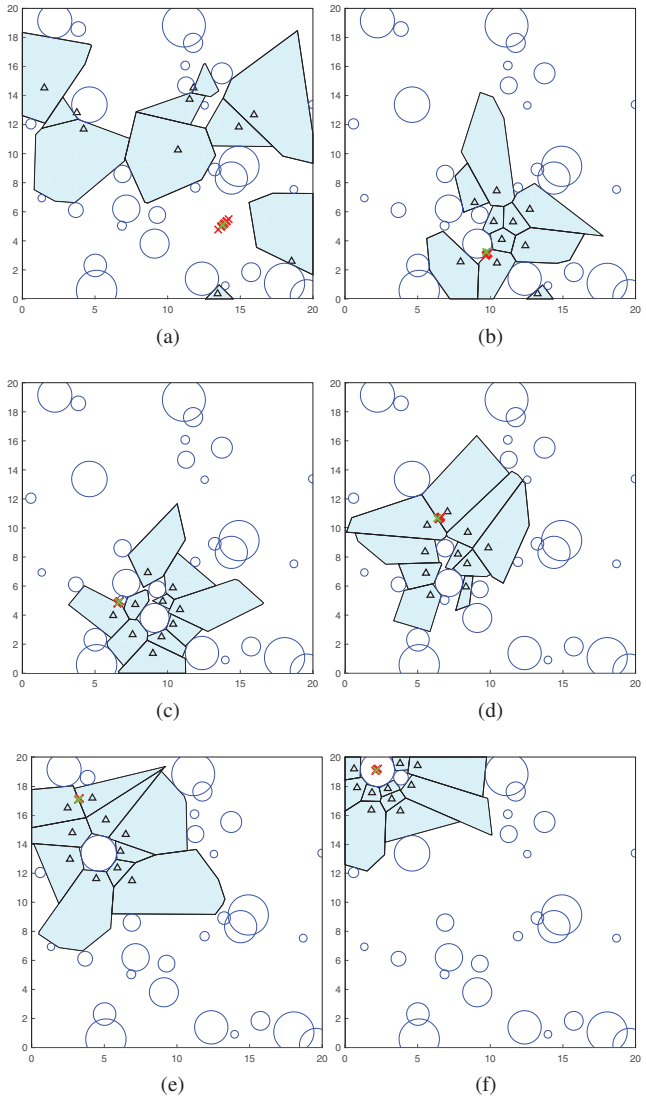


Fig. 3: Simulation of target tracking for $n = 10$ pursuers in an environment with $m = 30$ obstacles. The shaded regions indicate safe zones for the pursuers, and the target's position is denoted with the green x. Over time, all agents cluster around the target and track it as it moves.

agents are able to track the target throughout the environment and reduce the overall configuration cost.

B. Limiting Case

While our algorithm provides strong guarantees against collision avoidance, if a path from the agents to the target does not exist, agents may not rendezvous with the target. To illustrate this limitation, we present a simulation with $n = 7$ agents and $m = 43$ obstacles. Figure 6 shows the trajectories of the target and agents over time. As shown, a row of obstacles creates a barricade within the environment. The agents cluster near the obstacle boundaries, but without a connecting path between the obstacles, the agents cannot move to the location of the target.

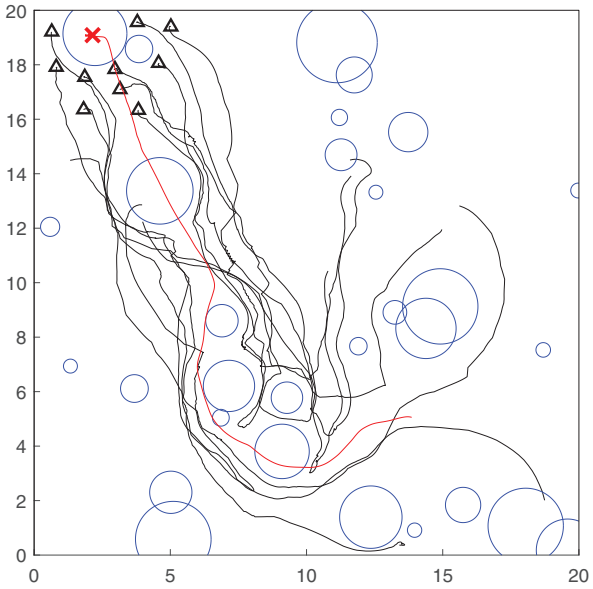


Fig. 4: Trajectories of all agents and target over time. The agents successfully track the target over time while simultaneously avoiding obstacles.

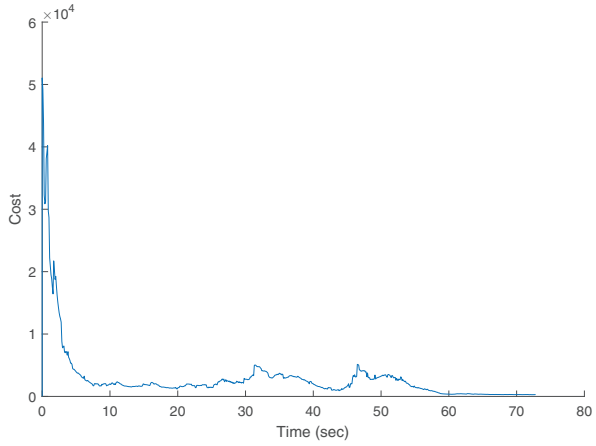


Fig. 5: Configuration cost over time. As the agents move closer, the configuration cost decreases. Local spikes in the cost function are due to the target moving through an obstacle, allowing it to increase the distance between itself and the agents. Despite local increases, the cost decreases again once the target is in free space.

V. Conclusion

In this paper, we present a framework for distributed, online target tracking through a cluttered environment. All agents are tasked with tracking a target while avoiding obstacles within the environment. We introduce an "Obstacle-Aware Voronoi Cell" (OAVC), which provides a safe-region for each agent to plan their control actions. The OAVC uses dynamics weights between each agent and obstacles to guarantee an agent's shared Voronoi boundary is tangent to the obstacle without intersecting it. Using this OAVC, we show the agents can successfully track their target using

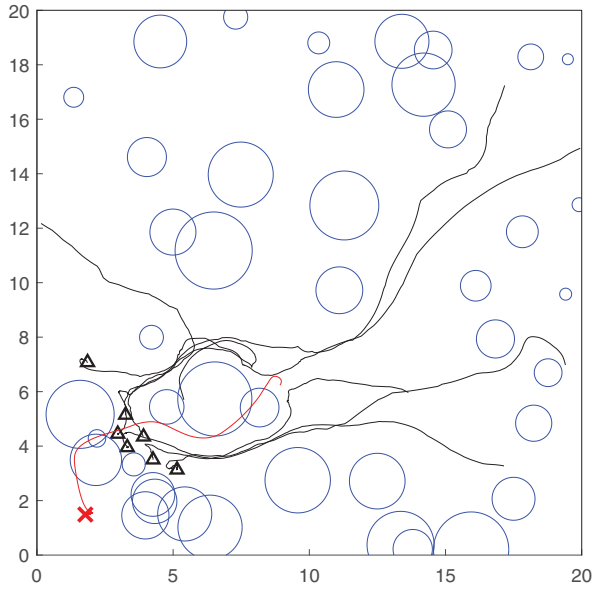


Fig. 6: Example scenario where the agents are unable to track the target. Here, the obstacles in the environment create a barricade that the agents cannot navigate around.

a move-to-centroid controller. This provides a distributed control law for all agents, simultaneously moving the agents closer to the target while guaranteeing collision avoidance. As an example of using an estimated target position, we use the example of generating position estimates with a Kalman filter and near-constant velocity model, demonstrating that the agents do not need perfect information about the target. Simulations in Matlab illustrate the effectiveness of our controller for a group of agents tracking a moving target. In our formulation, we place no restrictions on how the obstacles are placed in the environment. We present a "soft fail" case where the obstacles create a barricade in the environment, demonstrating a current limitation of our controller and an avenue for future research.

References

[1] 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.

[2] J.-C. Latombe, *Robot Motion Planning*, ser. The Springer International Series in Engineering and Computer Science. Springer, 1991.

[3] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, pp. 65–100, 2010.

[4] C. Schlegel, "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1998, pp. 594–599.

[5] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 341–346.

[6] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 1, pp. 45–59, 2004.

[7] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *Robotics, IEEE Transactions on*, vol. 21, no. 2, pp. 188–195, 2005.

[8] A. Ataei and I. C. Paschalidis, "Quadrotor deployment for emergency response in smart cities: A robust mpc approach," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 5130–5135.

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

[10] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," *Automatic Control, IEEE Transactions on*, vol. 55, no. 3, pp. 749–754, 2010.

[11] L. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. Mesquita, and G. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Algorithmic Foundation of Robotics VIII*, ser. Springer Tracts in Advanced Robotics, G. Chirikjian, H. Choset, M. Morales, and T. Murphey, Eds. Springer Berlin Heidelberg, 2010, vol. 57, pp. 85–99.

[12] S. G. Lee and M. Egerstedt, "Controlled coverage using time-varying density functions," in *Proc. of the IFAC Workshop on Estimation and Control of Networked Systems*, 2013.

[13] H. Huang, W. Zhang, J. Ding, D. Stipanovic, and C. Tomlin, "Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, Dec 2011, pp. 4835–4840.

[14] S. Pan, H. Huang, J. Ding, W. Zhang, D. Stipanovic, and C. Tomlin, "Pursuit, evasion and defense in the plane," in *American Control Conference (ACC), 2012*, June 2012, pp. 4167–4173.

[15] S.-Y. Liu, Z. Zhou, C. Tomlin, and K. Hedrick, "Evasion as a team against a faster pursuer," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5368–5373.

[16] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanovic, and C. J. Tomlin, "Cooperative pursuit with voronoi partitions," *Automatica*, vol. 72, pp. 64 – 72, 2016.

[17] A. Pierson, Z. Wang, and M. Schwager, "Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 530–537, April 2017.

[18] S. Bandyopadhyay, S. J. Chung, and F. Y. Hadaegh, "Probabilistic swarm guidance using optimal transport," in *2014 IEEE Conference on Control Applications (CCA)*, Oct 2014, pp. 498–505.

[19] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, April 2017.

[20] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[21] Z. Drezner, *Facility location: a survey of applications and methods*, ser. Springer series in operations research. Springer, 1995.

[22] S. Lee, Y. Diaz-Mercado, and M. Egerstedt, "Multirobot control using time-varying density functions," *Robotics, IEEE Transactions on*, vol. 31, no. 2, pp. 489–493, April 2015.

[23] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[24] Y. Bar-Shalom and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2001.