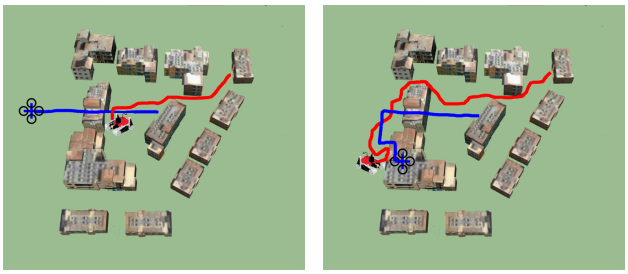
Tree Search Techniques for Adversarial Target Tracking with Distance-Dependent Measurement Noise

Zhongshun Zhang and Pratap Tokekar

***Abstract*—We study the problem of devising a closed-loop strategy to control the position of a robot that is tracking a possibly moving target. The robot obtains noisy measurements of the target's position. The measurement noise depends on the relative states of the robot and the target. We consider scenarios where the measurement values are chosen by an adversary so as to maximize the estimation error. Furthermore, the target may be actively evading the robot. Our main contribution is to devise a closed-loop control policy for distance-dependent target tracking that plans for a sequence of control actions, instead of acting greedily. We consider a game-theoretic formulation of the problem and seek to minimize the maximum uncertainty (trace of the posterior covariance matrix) over all possible measurement values. We exploit the structural properties of a Kalman Filter to build a policy tree that is orders of magnitude smaller than naive enumeration while still preserving optimality guarantees. We show how to obtain even more computational savings by relaxing the optimality guarantees. The resulting algorithms are evaluated through simulations and experiments with real robots.**



(a) Greedy. (b) Minimax.

Fig. 1. A ground robot tracks a target aerial robot. The ground robot cannot move through the obstacles, whereas the aerial robot can fly over them. In (a), the ground robot plans greedily and gets stuck behind obstacles. In (b), the minimax plans non-myopically by predicting the target's adversarial moves and is able to plan around the obstacles.

I. Introduction

Tracking a moving, possibly adversarial target is a fundamental problem in robotics and has long been a subject of study [4], [19]. Target tracking finds applications in many areas such as surveillance [24], telepresence [15], assisted living [20], and habitat monitoring [13], [28]. Target tracking refers to broadly two classes of problems: (i) estimating the position of the target using noisy sensor measurements; and (ii) actively controlling the sensor position to improve the performance of the estimator. The second problem is distinguished as *active* target tracking and is the subject of study of this paper.

One of the main challenges in tracking is that the target can be adversarial and actively avoid tracking by moving away from the robot. Furthermore, the value of future measurement locations can be a function of the unknown target state. Take as an example, a simple instance of estimating the unknown position of a stationary target where the measurement noise is a function of the distance between the robot and the target. If the true location of the target were known, the robot would always choose a control sequence that drives it closer to the target. Since, in practice, the true target location is unknown, we cannot determine such a control sequence exactly. A possible strategy, in this case, would be to plan with respect to the probability distribution of the target. However, the probability distribution itself will evolve as a function of the actual measurement values. This becomes even more challenging if the target is mobile.

In this paper, we use an Kalman Filter (KF) to estimate the state of a moving target with a possibly distance-dependent measurement model where the standard deviation of the measurement noise is a function of the distance between the robot and the target. Although it is common to assume that the noise is independent of the state, many sensors such as infrared [6] and radio-based ranging [12] exhibit distance-dependent noise. Distance-dependent noise models have been used for planning, for example, for achieving distributed consensus [21]. We focus on the problem of tracking a single target by planning the motion of a mobile robot.

There has been recent work on resilient and robust game-theoretic algorithms in adversarial settings [1], [26], [35]. We formulate the active adversarial target tracking problem as a minimax game. When planning non-myopically (for multiple steps in the future), one can enumerate all possible future measurements in the form of a tree. In particular, a minimax tree can be used to find the optimal (in the minimax sense) control policy for actively tracking a target [30]. The size of the minimax tree grows exponentially with the time horizon. The tree can be pruned using alpha-beta pruning [25]. Our main contribution is to show how the properties of an Kalman filter can be exploited to prune a more significant number of nodes without losing optimality. In doing so, we extend

The authors are with the Department of Computer Science, University of Maryland, College Park, Maryland, USA. {zszhang,tokekar}@umd.edu.

This material is based upon work supported by the National Science Foundation under Grant #1566247.

the pruning techniques first proposed by Vitus et al. [33] for linear systems with state independent noise. Using a minimax tree, we generalize these results to a system with distance-dependent noise. Our pruning techniques allow us to trade-off the size of the tree (equivalently, computation time) with the optimality guarantees of the algorithm. We demonstrate this effect in simulations and proof-of-concept indoor and outdoor tracking experiments. We also show how the tree can be updated online, when the measurements received and/or the target motion is not adversarial.

The rest of the paper is organized as follows. We start with the related work in Section II. The problem formulation is presented in Section III. Our main algorithm is presented in Section IV. The pruning condition in different cases is discussed in Section IV. We validate the algorithm through simulations that are described in Section V and experiments on real robots reported in Section VI. Finally, we conclude with a brief discussion of future work in Section VII.

A preliminary version of the paper first appeared in IEEE Conference on Decision and Control 2016 [38]. This journal version is an extension of the conference paper, including detailed proofs, the online execution algorithm, the Gazebo simulations, and physical experiments with the ground and aerial robots.

II. Related Work

The target tracking problem has been studied in various settings. Bar-Shalom et al. [4] have presented many of the commonly-used estimation techniques in target tracking. The five-part survey by Li and Jilkov [19] covers commonly-used control and maneuvering techniques for active target tracking.

Pursuit-evasion is a class of problems typically used to study adversarial target tracking [9]. Kolling and Carpin [16] study a pursuit-evasion problem on graphs that model the detection of intruders in complex indoor environments by robot teams. A typical approach is to model the problem as a non-cooperative game and use Pontryagins minimum principle and the Bellman equation to find the optimal paths for the pursuer [5], [18]. These approaches typically assume noise-free sensing. Amongst pursuit-evasion works that explicitly address measurement noise are works by Vander Hook and Isler [32] using noisy bearing sensors.

Willman [34] studied the differential pursuit-evasion problem with state-independent Gaussian noise to the motion model as well as the measurement model. The author showed that the problem can be reduced to a deterministic one. Yaesh and Shaked [36] have shown the connection between the H_∞ optimal estimation theory and adversarial target tracking. Zengin and Dogan [37] have presented a real-time target tracking for autonomous UAVs in adversarial environments. More recently, Gu [11] proposed a minimax filter to estimate the state of an adversarial target using noisy measurements from a sensor network. Specifically, they modeled the estimation problem as a differential zero-sum game. Unlike these works, we use a minimax search tree to (non-myopically) plan for the control actions of the robot. Karaman et al. [14] showed how to solve a similar pursuit-evasion problem using RRT*. However, unlike the problem we study, their problem setup has no notion of measuring the target's (or the other agent's) position. As such, there is no need to run a state estimator and no way to handle distance dependent measurement noise as we do in this paper.

A search tree can provide optimal or near-optimal policies for target tracking. Li et al. [22] solve a visibility-based pursuit-evasion problem using tree search techniques. However, building a search tree can be computationally expensive especially for large-scale instances. The key is to prune the tree to yield computational savings effectively. Vitus et al. [33] presented an algorithm that computes the optimal scheduling of measurements for a linear dynamical system. The goal is to track a linear dynamical system using a set of sensors such that one sensor can be activated at any time instance. The posterior covariance in estimating a linear system in a Kalman filter depends on the prior covariance and sensor variance but not on the future measurement values (unlike the case in non-linear systems). Thus, one can build a search tree enumerating all possible sensor selections and choosing the one that minimizes the final covariance. The main contribution of Vitus et al. is to present a pruning technique to reduce the size of the tree while still preserving optimality.

Monte-Carlo Tree Search (MCTS) [25], is an alternative algorithm to solve these discrete, two-player, zero-sum games. MCTS has been shown to be more effective in solving large two-player games, such as Go [10]. However, Li et al. [22] show that minimax has the advantage in finding deterministic solutions compare to MCTS in a pursuit-evasion game. Nevertheless, the pruning idea proposed in this paper can also be applied in MCTS to remove redundant nodes.

Atanasov et al. [3] extended this result to active target tracking with a single robot. A major contribution was to show that robot trajectories that are nearby in space can be pruned away (under certain conditions), leading to further computational savings. This was based on a linear system assumption. In this paper, we build on these works and make progress towards generalizing the solution for distance-dependent observation systems.

A major bottleneck for planning for distance-dependent observation systems is that the future covariance is no longer independent of the actual measurement values. In Kalman filtering, the covariance update equation contains the noise variance, which in our case, depends on the position of the target. Since we do not know the actual position of the target, we use the estimated position which depends on the measurements. Thus, the search tree will have to include possible measurement values. Furthermore, finding an optimal path is no longer sufficient. Instead, one must find an optimal policy that prescribes the optimal set of actions for all possible measurements. We show how to use a minimax tree to find such an optimal policy while at the same time leveraging the computational savings that hold for the linear case.

III. Problem Formulation

We assume that the position of the robot is known accurately using onboard sensors (*e.g.*, GPS, laser, IMU, cameras). The

motion model of the robot is given by:

$$X_r(t+1) = f(X_r(t), u_r(t)) \tag{1}$$

where, $f(\cdot)$ is the motion model of the robot, $X_r = [x_r(t), y_r(t)]^T \in \mathbb{R}^2$ is the position of robot and $u_r(t) \in \mathcal{U}_r(t)$ is the control input at time t.[1] $\mathcal{U}_r(t)$ is a finite space of control inputs. We assume there are n actions available as control inputs[2]:

$$\mathcal{U}_r(t) = \{u_{r1}(t), u_{r2}(t), \cdots, u_{rn}(t)\}.$$

The robot has a sensor that gets a measurement of the target's position. We assume that the target's motion model is given by:

$$X_o(t+1) = C_t X_o(t) + u_o(t) + v(t) \tag{2}$$

where, $X_o(t) = [x_o(t), y_o(t)]^T$ is the position of the target and $v(t)$ is the process noise drawn from a Gaussian distribution of known covariance. Here, $u_o(t) \in \mathcal{U}_o(t)$ is the control input for the target at time t with n' actions available inputs:

$$\mathcal{U}_o(t) = \{u_{o1}(t), u_{o2}(t), \cdots, u_{on'}(t)\}.$$

The task of the robot is to track the target using its noisy measurements. The measurements, $z(t)$, can be a function of the states of the target and the robot:

$$z(t) = HX_o(t) + w(X_r(t), X_o(t)) \tag{3}$$

The measurement noise, $w(t) \sim \mathcal{N}(0, \Sigma_w(t))$, is drawn from a 2D zero-mean Gaussian distribution. Specifically, we have

$$\Sigma_w(t) = \begin{bmatrix} \delta_1^2 + \delta_2^2 d(X_r(t), X_o(t)) & 0 \\ 0 & \delta_1^2 + \delta_2^2 d(X_r(t), X_o(t)) \end{bmatrix}.$$

Note that the noise variance $\Sigma_w(t)$ along each dimension is independent and given by $\delta_1^2 + \delta_2^2 d(X_r(t), X_o(t))$, where δ_1 and δ_2 are positive constant values. In general, the noise variance along the two dimensions can use different δ_1, δ_2 values; we use the same δ_1 and δ_2 values for both dimensions for ease of exposition. Here, $d(X_r(t), X_o(t)) =$

$$\begin{cases} \mathcal{C}, & ||X_r(t) - X_o(t)||_2 > \mathcal{B} \\ \dfrac{\mathcal{C}||X_r(t) - X_o(t)||_2}{\mathcal{B}}, & ||X_r(t) - X_o(t)||_2 \leq \mathcal{B}. \end{cases} \tag{4}$$

Here, $||\cdot||_2$ is the 2-norm which corresponds to the Euclidean distance between the robot and the target.

When the true distance between the robot and target is within $\mathcal{B}$, we assume that measurement noise variance is proportional to the true distance. When the true distance is greater than $\mathcal{B}$, the variance saturates at the maximum constant value of $\mathcal{C}$. Here, we consider a linear measurement model. A more general case is that the measurement model can be a non-linear function of the robot and target's state. For that case, we can use the Extended Kalman Filter and compute the observation matrix H by using the Jacobian of the measurement function [8].

[1]This result can be extended to 3D and other state spaces

[2]Note that the subscript r denotes terms related to the robot and subscript o denotes the terms related to the target (not to be confused with the time). The time is always denoted inside paranthesis (t).

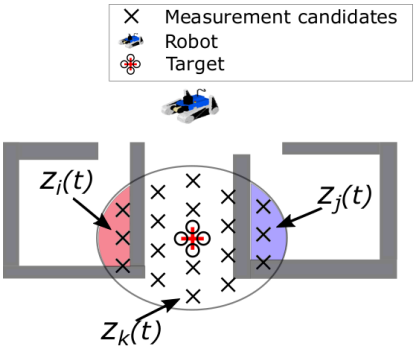


Fig. 2. If the robot obtains a noisier measurement (say $z_i(t)$) at time t, a greedy strategy may follow the mean of the estimated target's position and get stuck in a room (left in the case of $z_i(t)$ even if the actual target is outside the room. A minimax strategy that plans against worst-case measurements for a finite horizon will instead hedge the bets, and stay in the middle until the target's estimate shrinks enough.

The estimated position and the covariance matrix of the target at time t, $\hat{X}_o(t)$ and $\hat{\Sigma}_o(t)$, are given by the Kalman filter. The uncertainty in the estimate of the target's position is measured by the trace of the covariance matrix. The goal of the robot is to reduce the uncertainty in the target's estimate. The goal of the adversary is to increase the uncertainty. There are two types of adversaries:

1) **Game against an escaping target:** Here, the target actively chooses $u_o(t)$ to increase the estimation error. In our model, since the measurement noise is a function of the distance between the robot and the target, an adversarial target will choose to move away from the robot. When there are obstacles present in the environment, a myopic strategy that follows the estimated target position may get stuck (see Figure 1). Instead, we use minimax strategy that is able to track an escaping target better using a non-myopic approach.
2) **Game against nature:** Here, "nature" acts as an adversary and generates the worst-case measurements, $z(t)$, for the robot. In general, the measurements will not be the worst-case ones. However, planning against this adversary will lead to a policy that is robust to possible outlier measurements. Consider Figure 2 where the actual target is located in the corridor between the rooms. If at time t, the robot obtains an outlier measurement (say $z_i(t)$), the estimated target position will shift closer to the left room. A myopic planner will choose to move left and may get stuck in the room. On the other hand, a minimax planner that considers adversarial measurements will choose to stay in the middle until such time that the estimated target covariance shrinks sufficiently. This way, even though the actual measurements are drawn from a random distribution, we can be robust to the noise by considering a game against nature.

The minimax tree strategy can be applied to either or both types of adversaries. Formally, the problem considered in this paper can be formally stated as follows.

Problem 1. *Given an initial deterministic robot position, $X_r(0)$, and an initial target estimate, $[\hat{X}_o(0), \hat{\Sigma}_o(0)]$,*

find a sequence of control laws for the robot, $\sigma = u_r(0), u_r(1), \cdots, u_r(T)$ *from time* $t = 0$ *to* $t = T$ ($u_r(i) \in \mathcal{U}_r(i)$) *to minimize trace of the covariance in the target's estimate at time* $t = T$. *That is,*

$$\min_{u_r(0),\ldots,u_r(T)} \quad \max_{\substack{u_o(0),\ldots,u_o(T);\\ z(0)\ldots,z(T)}} \quad tr(\Sigma_T). \tag{5}$$

such that,

$$\Sigma_{t+1} = \rho_t(\Sigma_t), \quad t = 0, 1, \cdots, T-1$$

where $\rho_t(\cdot)$ *is the Kalman Riccati equation [17].*

The Kalman Riccati equation [17], $\rho_t(\cdot)$, maps the current covariance matrix $\hat{\Sigma}_t$ to the covariance matrix at $t+1$ using the measurement $z(t)$:

$$\rho_t(\hat{\Sigma}_t) = C_t\hat{\Sigma}_t C_t^T - C_t\hat{\Sigma}_t H^T(H\hat{\Sigma}_t H^T + \Sigma_w)^{-1}H\hat{\Sigma}_t C_t^T + \Sigma_v. \tag{6}$$

Note that we solve Problem 1 at each time instance looking ahead T timesteps. Even though the solution for Problem 1 is a sequence of control inputs, $u_r(0), \ldots, u_r(T)$, we only apply the first one $u_r(0)$. Then, we use the actual measurement $z(0)$ to update the covariance matrix and then solve Problem 1 again for T timesteps. This is further explain in Section IV-D.

The true position of the target is unknown making it impossible to determine Σ_w exactly. Consequently, we use an estimate of Σ_w, denoted by $\hat{\Sigma}_w$, using the estimated target's position, $\hat{X}_o(k)$. Specifically, $\hat{\Sigma}_w$ is obtained by replacing $X_o(t)$ with $\hat{X}_o(k)$ in Equations (3)–(4). Since $\hat{X}_o(t)$ is computed using the values of $u_o(t)$ and $z(t)$, the Kalman Riccati map using the estimated noise covariance, $\hat{\Sigma}_w$, becomes a function of $u_o(t)$ and $z(t)$,

$$\rho_t(\hat{\Sigma}_t) = C_t\hat{\Sigma}_t C_t^T - C_t\hat{\Sigma}_t H^T(H\hat{\Sigma}_t H^T + \hat{\Sigma}_w(u_o(t), z(t)))^{-1}H\hat{\Sigma}_t C_t^T + \Sigma_v. \tag{7}$$

Remark 1. *The intuition behind the objective function is that it seeks to minimize the estimation uncertainty in the worst case. Also, note that the covariance will decrease as the robot and the target get closer since we assume the variance in the measurement noise is dependent on the distance between the robot and the target.*

Remark 2. *We can choose other measures for the objective function, such as the determinant [3]. We use trace since it is more robust in the following sense. For a 2D covariance matrix, the eigenvalues are the lengths of the major and minor axis of the uncertainty ellipse. If one of the two eigenvalues of the covariance matrix is close to* 0*, the determinant will also be close to* 0*, even if the other eigenvalue (i.e., the uncertainty along that direction) is very large. On the other hand, the trace will be large, if one of the two eigenvalues is large.*

Earlier works [3], [33] solve a similar problem but with a linear Gaussian system. The linearity assumption makes the Riccati equation independent of the position of the target (known as the separation principle). Consequently, they show an open loop policy can determine the optimal control sequence for the robot. In our case, the optimal control policy will be a closed-loop one since the measurement noise is a function of the position of the target. However, this generalization comes at the expense of discretization of the set of possible target measurements.

Equation (7) shows that trace at time T will be a function of $u_o(t)$ and $z(t)$. When we plan for a finite horizon, we do not know the exact sequence of actions the target takes, $u_o(t)$, and true measurements, $z(t)$, that we obtain. Instead, we enumerate all possible actions the target can take and all possible measurements that we may obtain. Since the domain of the measurements is infinite, we discretize and assume that the measurement at any time step is chosen from one of m tuples of candidate measurements.[3] That is,

$$\hat{Z}(t) \in \{z_1(t), z_2(t), \cdots, z_m(t)\}. \tag{8}$$

For example, we can choose m candidate measurements from the data within 3 standard deviations of the mean value, which contain 99.7% of the possible measurements. Figure 3 shows a discretized Gaussian distribution.

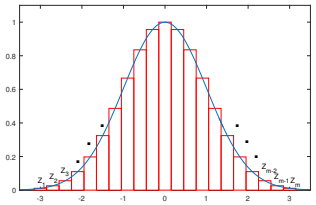


Fig. 3. We discretize the set of candidate measurements using m samples drawn from a Gaussian distribution.

Note that the enumerated possible measurements are estimated while we build the search tree. To clarify the difference, we use the following notation:

- $\hat{Z}(t) = \{z_1(t), \ldots, z_i(t), \ldots, z_m(t)\}$: set of possible measurement that is obtained at time t.
- $z_i(t)$: possible measurement at time t.
- $z(t)$: actual measurement obtained at time t. Here, $z(t)$ may or may not be the worst-case measurement.

In the next section, we present the minimax tree strategy and various pruning techniques that allow us to efficiently find the optimal closed-loop policy for this problem.

IV. The Minimax Algorithm and Pruning Techniques

If the trajectory of the target is known, then we can find the optimal path for the robot using techniques such as dynamic programming. However, when tracking an adversarial target, we need to use game-theoretic planning to find the policy for the robot. We model Problem 1 as a sequential game[4] played between the robot and an adversary. The robot executes a control action, takes a measurement of the target, the target moves to the new position, and the sequence repeats. Based on this, we generate a search tree to find the optimal policy.

[3]These m candidate measurements can be obtained by, for example, sampling from the continuous distribution of zero mean sensor noise around the current estimate of the target.

[4]Note that, in practice, the robot and the target can move simultaneously.

TABLE I
SEARCH TREE AND PRUNING TECHNIQUES FOR VARIOUS MODELS

Measurement Noise	Target Motion	Strategy	Pruning Algorithm
State-independent	Known/Stationary	Search tree	Algebraic redundancy (Theorem 1)
State-independent	Adversarial	Minimax tree	Alpha-beta pruning, Algebraic redundancy (Theorem 1)
Distance-dependent	Known/Stationary	Minimax tree	Alpha pruning
Distance-dependent	Adversarial	Minimax tree	Alpha-beta pruning, Algebraic redundancy (Theorem 2)

The adversary (*i.e.,* nature) chooses measurement noise and the target chooses actions at every step, whereas the robot chooses its own actions. By optimizing the minimax trace, the robot determines the best conservative policy.

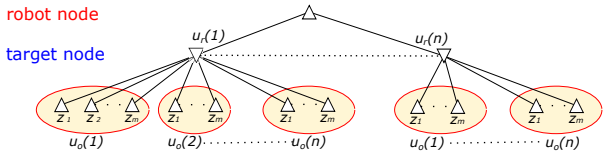


Fig. 4. One step minimax tree enumerates all the possible actions the robot and target as well as all candidate measurements. Each node in the tree stores the robot position and estimated target position.

We find this optimal strategy by building a minimax tree. Figure 4 shows an example. This tree enumerates all possible control laws for the robot and the target and all possible measurements that the robot can obtain. A node on the kth level of the tree stores the position of the robot, $X_r(k)$, the estimated position of the target, $\hat{X}_o(k)$, and the covariance matrix $\hat{\Sigma}_k$. Each node at an odd level has one branch per control action of the robot. We term these nodes as "robot nodes." Each node at an even level has one branch per tuple of candidate measurement and candidate actions for the target. These nodes are termed as the "target nodes."

The robot's state and the target's estimate are updated appropriately along the control and measurement branches using the state transition equation (Equation 1) and the Kalman filter update equation, respectively. The minimax value is computed at the leaf nodes and is equal to the trace of the covariance matrix at that node. These values are propagated upwards to compute the optimal strategy.

The full enumeration tree has a size exponential in the number of control actions, candidate measurements, and the planning horizon. We present three pruning strategies to reduce the size of the tree. The first two are based on existing alpha-beta pruning while the third one is a new contribution of this paper. Table I gives the summary of all pruning techniques.

A. Alpha-Beta Pruning

As a first step in reducing the size of the tree, we use alpha-beta pruning [25]. The main idea in alpha-beta pruning is that if we have explored a part of the tree, we have an upper or lower bound on the optimal minimax value. For example, in alpha pruning where we consider the upper bound, when exploring a new node, n_i, if we find that the minimax value of the subtree rooted at n_i is greater than the upper bound found, that subtree does not need to be explored further. This is because an optimal strategy will never prefer a strategy that passes through n_i since there exists a better control policy in another part of the tree. Note that n_i must be a robot node. Figure 5 shows an example of alpha-beta pruning. Target nodes cannot be pruned since the robot has no control over the actual measurement values. That is, we only apply alpha-pruning. When the measurements and target's motion is known to be adversarial, then full alpha-beta pruning can be applied. In such a case, the target nodes can be pruned away as well.

B. Algebraic Redundancy Pruning

Vitus et al. [33] presented algebraic redundancy pruning of search trees (not minimax trees) for linear systems with state-independent noise. The key idea is that if a node, n_i, has higher uncertainty than another node, n_j at the same level, then any descendant of n_i will always have higher uncertainty than some descendant of n_j. Therefore, n_i is redundant and can be pruned away. They use the monotonicity and concavity of the Riccati mapping in linear systems with state-independent noise to prove the following result.

Theorem 1 (Algebraic Redundancy [33])**.** *Let $\mathcal{H} = \{(X_r^i(t), \Sigma_t^i)\}$ be a set of n nodes at the same level of the tree. If there exist non-negative constants $\alpha_1, \alpha_2, \ldots, \alpha_k$ such that,*

$$\Sigma_t^p \succeq \sum_{i=1}^{k} \alpha_j \Sigma_t^j \quad \textit{and} \quad \sum_{i=1}^{k} \alpha_i = 1$$

then the node $(X_r^p(t), \Sigma_t^p)$ is regarded as algebraically redundant[5] with respect to $\mathcal{H} \setminus \{(X_r^p(t), \Sigma_t^p)\}$ and $(X_r^p(t), \Sigma_t^p)$ and all of its descendants can be pruned without eliminating the optimal solution from the tree.

They prove that the trace of any successor of $(X_r^p(t), \Sigma^p(t))$ cannot be lower than one of the successors of $\mathcal{H} \setminus \{(X_r^p(t), \Sigma^p(t))\}$. Our main insight is that, a similar redundancy constrained can be defined for the non-linear case with suitable additional constraints as described below.

We extend these ideas for minimax trees with possibly distance-dependent noise. We first prove the monotonicity of distance-dependent Riccati equation.

Lemma 1. *Let A and B be two nodes in the same level of the minimax tree and Σ_t^A, S^A and Σ_t^B, S^B be the corresponding*

[5] $M \succeq N$ represents that $M - N$ is positive semi-definite.

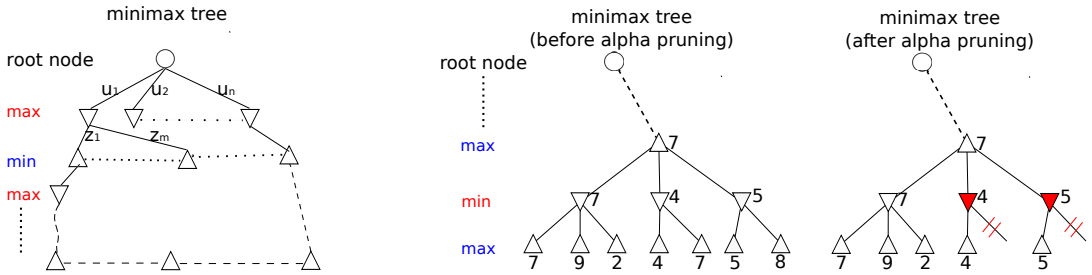


Fig. 5. A minimax tree with alpha pruning. $\bigtriangledown$ and $\bigtriangleup$ are nodes in which we compute the minimum or maximum value of its children. The value at the leaf nodes equals the $\mathrm{tr}(\Sigma_k)$. $\bigtriangledown$ and $\bigtriangleup$ nodes represent robot and target nodes, respectively. The filled $\bigtriangledown$ are pruned by alpha pruning.

covariance matrices and measurement noise covariance matrices. If $\Sigma_t^A \succeq \Sigma_t^B$ and $S^A \succeq S^B$, then after applying one step of the Riccati equation, we have $\rho(\Sigma_t^A) \succeq \rho(\Sigma_t^B)$.

Note that from the definition of the covariance matrix, the matrices S^A, S^B are positive definite, $S^A \succeq 0$, $S^B \succeq 0$.

We will show conditions under which a node A is redundant with respect to a set of nodes, termed as candidate set, that is already present in the tree. Figure 6 shows an example.

Definition 1. *Let $\mathcal{H}$ be a set of nodes. The set $\mathcal{H}$ is called as a* candidate set *with respect to some node A if (i) the nodes in $\mathcal{H}$ are at the same level as that of A; and (ii) every node $B \in \mathcal{H}$ is on the optimal minimax path (the highlighted path in Figure 6) for the subtree rooted at the least common ancestor of A and B.*

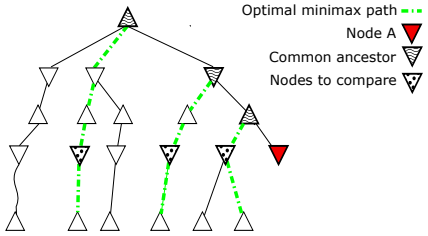


Fig. 6. Nodes in the candidate set, $\mathcal{H} = \{(X_r^i(t), \hat{X}_o^i(t), \hat{\Sigma}_t^i)\}$, of node A are marked as red. Their least common ancestor is highlighted.

Before we present the full details, we list the conditions that will be used in Theorem 2. We have more conditions since pruning with distance-dependent noise is a general version of Theorem 1.

Let $\mathcal{H} = \{(X_r^i(t), \hat{X}_o^i(t), \hat{\Sigma}_t^i)\}$ be the candidate set of N nodes with respect to some node $A = (X_r^A(t), \hat{X}_o^A(t), \hat{\Sigma}_t^A)$. The conditions are as follows:

(C1) the robot and estimated target states are identical, *i.e.,* $X_r^A(t) = X_r^i(t)$ and $\hat{X}_o^A(t) = \hat{X}_o^i(t)$ for all i in $\mathcal{H}$;[6]

(C2) the least common ancestor of A with any other node in $\mathcal{H}$ is a MIN (robot) node;

(C3) the least common ancestor of A with any other node in $\mathcal{H}$ is a MAX (target) node;

[6]For time-invariant linear systems with constant H and C, condition (C1) is not required since all the covariance matrices are updated through the same Kalman filter Riccati equations.

(C4) HH^T is invertible and there exist non-negative constants $\alpha_1, \alpha_2, \ldots, \alpha_N$ such that,

$$\Sigma_t^A \succeq \sum_{i=1}^{N} \alpha_i \Sigma_t^i \tag{9}$$

$$S_t^A \succeq \sum_{i=1}^{N} \alpha_j S_t^i + (T-t)\left(\delta_1^2 + \delta_2^2 \mathcal{C}\right) I_{2\times 2} \tag{10}$$

where $\sum_{i=1}^{N} \alpha_i = 1$.

(C5) HH^T is invertible and there exist non-negative constants $\alpha_1, \alpha_2, \ldots, \alpha_N$ such that,

$$\Sigma_t^A \preceq \sum_{i=1}^{N} \alpha_i \Sigma_t^i \tag{11}$$

$$S_t^A \preceq \sum_{i=1}^{N} \alpha_j S_t^i + (T-t)\left(\delta_1^2 + \delta_2^2 \mathcal{C}\right) I_{2\times 2} \tag{12}$$

Our main result is stated as follows.

Theorem 2. *[Distance-dependent Algebraic Redundancy] Let $\mathcal{H} = \{(X_r^i(t), \hat{X}_o^i(t), \hat{\Sigma}_t^i)\}$ be the candidate set of N nodes with respect to some node $A = (X_r^A(t), \hat{X}_o^A(t), \hat{\Sigma}_t^A)$. If the node A satisfies (C1), (C2) and (C4), then there exists a node in $\mathcal{H}$, say B, such that:*

$$tr(\Sigma_T^A) \geq tr(\Sigma_T^B).$$

Similarly, if node $A = (X_r^A(t), \hat{X}_o^A(t), \hat{\Sigma}_t^A)$ satisfies (C1), (C3) and (C5), then there exists a node in $\mathcal{H}$, say B, such that:

$$tr(\Sigma_T^A) \leq tr(\Sigma_T^B).$$

In both cases, the node A can be pruned from the minimax tree without affecting the optimal policy.

These conditions are based on the algebraic redundancy conditions given in [33] (Theorem 1) for the sensor scheduling problem. The search tree in [33] is a non-adversarial search tree whereas we generalize these conditions to the adversarial case using a minimax search tree with two types of nodes (MIN and MAX).

The conditions in Theorem 2 state when a node A can be made redundant by another node B. The two nodes must have the same robot and estimated target states as given in (C1). In informal terms, Theorem 2 states that if node B is better than node A, some descendant of node B will be better than that of

node A. As a result, node A can be pruned from the tree. The notion of "better" depends on whether their common ancestor is a MIN (robot controlled) or a MAX (target controlled) node, given in (C2) and (C3). If their common ancestor is a MIN node, then we say B is better if it has lower uncertainty than A. In this case, the robot will never choose the path that leads to A as opposed to B. If the common ancestor is a max node, then we say B is better if it has higher uncertainty than A. As a result, the target will always prefer the path that leads to B than A. In both cases, the optimal path will not include A which can, therefore, be pruned away.

We use the above result to prune away nodes while building the tree. Note that the algebraic redundancy pruning is more effective when the tree is being built in a breadth-first fashion (since we compare nodes at the same level). On the other hand, the alpha-beta pruning is useful only when the tree is built in a depth-first fashion. In order to apply both pruning strategies, the tree must be built depth-first. While adding a new node to the tree, we check whether the conditions in Theorem 2 are satisfied with respect to all other existing nodes at the same level. In order to check for the optimal path, we require at least one path to a leaf node from the current node. Therefore, the conditions can be checked for all predecessors of the current node under consideration. If the conditions are met for any predecessor, then the predecessor node (and all its descendants) are pruned from the tree.

Since $\mathcal{H}$ can be of any size, checking for conditions (C4) and (C5) can be computationally expensive and requires solving a Linear Matrix Inequality (LMI). If we restrict $\mathcal{H}$ to contain only one node, then conditions (C4) and (C5) amount to checking positive semidefiniteness of a matrix which can be done much faster. However, this would mean fewer nodes get pruned away. As the level of the tree grows, the number of nodes increases exponentially. We can trade-off the two factors, by solving LMI at lower depths in the tree (when there are fewer nodes), and then only making pairwise comparisons for higher depths. We provide a more detailed discussion of the steps we implement to check (C1)–(C5) in the simulation section (Section V). In the next section, we describe more ways of saving computational time at the expense of optimality.

C. Sub-optimal Pruning algorithm

We can further reduce the number of branches at the expense of losing optimality by relaxing the alpha-pruning and algebraic redundancy constraints. We use two parameters $\epsilon_1 > 0$ and $\epsilon_2 > 0$ as relaxation parameters for alpha pruning and algebraic redundancy pruning, respectively. In each case, we bound the loss in optimality as a function of the parameters.

Specifically, while building the tree, we prune away a node if it satisfies either of the following two conditions. When checking for alpha pruning, we prune a node if its alpha value is greater than or equal to the best minimax value found so far minus ϵ_1. Similarly, we replace the first constraint of (C4) (Equation (9)) in Theorem 2 with the following:

$$\Sigma_t^A + \epsilon_2 \succeq \sum_{i=1}^{N} \alpha_i \Sigma_t^i. \tag{13}$$

Similar condition can be applied to the first constraint of (C5) given in Equation (10) of Theorem 2.

By varying ϵ_1 and ϵ_2, we can vary the number of nodes in the search tree. Next we bound the resulting loss in the optimality of the algorithm.

Theorem 3 (ϵ_1-alpha pruning)**.** *Let $J^*_{2k} = \mathrm{tr}(\hat{\Sigma}^*_{2k})$ be the optimal minimax value returned by the full enumeration tree. If $J^{\epsilon_1}_{2k} = \mathrm{tr}(\hat{\Sigma}^{\epsilon_1}_{2k})$ is the value returned by the ϵ_1–alpha pruning algorithm, then $0 \leq J^{\epsilon_1}_{2k} - J^*_{2k} \leq \epsilon_1$.*

The proof follows directly from the fact that if a node on the optimal policy, say n_i is pruned away, then the alpha value at n_i is at most the alpha value of some other node, say n_j, that is present in the tree minus ϵ_1. The alpha value of n_j cannot be less than the value returned by the ϵ_1 algorithm. The bound for ϵ_2-algebraic redundancy pruning is more complicated.

Theorem 4 (ϵ_2-State dependent algebraic redundancy pruning)**.** *Let $J^*_{2k} = \mathrm{tr}(\hat{\Sigma}^*_{2k})$ be the optimal minimax value returned by the full enumeration tree of $2k$ levels. If $J^{\epsilon_2}_{2k} = \mathrm{tr}(\hat{\Sigma}^{\epsilon_2}_{2k})$ is the value returned by the ϵ_2–algebraic redundancy pruning algorithm, then*

$$0 \leq J^{\epsilon_2}_{2k} - J^*_{2k} \leq B^{\epsilon_2}$$

where,

$$B^{\epsilon_2} = \mathrm{tr}\left\{ \sum_{j=1}^{k} \left[\prod_{i=j-1}^{0} \left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right) \prod_{i=0}^{j-1} \left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)^T \right] \epsilon_2 \right\}$$

where, $F_i(\Sigma) = C - CK_i(\Sigma)H$ and $K_i(\Sigma)$ is the Kalman gain given by $K_i(\Sigma) = \Sigma H^T (H\Sigma H^T + \Sigma_w)^{-1}$, and $\Phi_{2k}(\cdot)$ is the application of the Riccati equation $\rho(\cdot)$, over k measurement steps:

$$\Phi_{2k}(\cdot) = \underbrace{\rho_{2(k-1)}(\rho_{2(k-2)}(\ldots \rho_0(\cdot)))}_{k \text{ steps } \rho(\cdot)}. \tag{14}$$

Note that in Theorem 4, the conditions (C1)–(C3) are still required. (C1) and (C2) are always required for the MAX level, (C1) and (C3) are required for the MIN level. Theorem 4 relaxes the condition for (C4) for the MAX level, and (C5) for the MIN level.

By combining the two results, let $J^{\epsilon_1,\epsilon_2}_{2k}$ be the upper bound when we apply ϵ_1-alpha pruning and ϵ_2-Distance dependent algebraic redundancy pruning together, we get

$$0 \leq J^{\epsilon_1,\epsilon_2}_{2k} - J^*_{2k} \leq \max\left\{\epsilon_1, B^{\epsilon_2}\right\}.$$

The parameter ϵ_1 has a direct relationship with the suboptimality. On the other hand, ϵ_2 has a more indirect relationship with the suboptimality. In the next section, we plot the upper bound of $J^{\epsilon_1}_{2k}$ and $J^{\epsilon_2}_{2k}$, given by Theorems 3 and 4, in order to visualize this relationship. The selection of ϵ_1 and ϵ_2 can balance the trade-off between the size of the tree (equivalently, computation time) with the optimality guarantees of the algorithm. The bound help us determine the extent of the trade-off.

Algorithm 1: The Minimax Search With Pruning.

function Minimax($node, depth, \alpha, \beta, state$)
  if *node i is a leaf node* **then**
    return $tr(\Sigma_T^i)$
  else if *state is at the MAX level* **then**
    $bestvalue \leftarrow -\infty$
    for *each control input* $u_{r1}(t), \cdots, u_{rn}(t)$ **do**
      $v \leftarrow$ New robot states
      $V \leftarrow \text{Minimax}(v, depth-1, \alpha, \beta, \text{MIN})$
      $bestvalue \leftarrow \max(bestvalue, V)$
      $\alpha \leftarrow \max(bestvalue, \alpha)$
      `// Alpha-beta pruning`
      if $\beta \leq \alpha$ **then**
        break
      end
      `// Check the condition in Theorem 2`
      if $(C1)\&(C2)\&(C4)$ *are true* **then**
        break
      end
      return *bestvalue*
    end
  else
    $bestvalue \leftarrow +\infty$
    for *each candidate measurements* $z_1(t), \cdots, z_m(t)$ **do**
      $v \leftarrow$ Update estimated target states
      $V \leftarrow \text{Minimax}(v, depth-1, \alpha, \beta, \text{MAX})$
      $bestvalue \leftarrow \min(bestvalue, V)$
      $\beta \leftarrow \min(bestvalue, \beta)$
      if $\beta \leq \alpha$ **then**
        break
      end
      if *(C1)&(C3)&(C5) are true* **then**
        break
      end
      return *bestvalue*
    end
  end
end
$\{S_0\} \leftarrow$ Initial
$\text{Minimax}(S_0, 1, -\infty, \infty, \text{MAX})$

D. Online Execution of the Search Tree

Once the tree is built, the robot can execute the optimal policy. At the root node, the robot executes the first control action along the optimal minimax path found. Then, the robot obtains a measurement. This measurement may not correspond to the worst-case measurement. Furthermore, the actual value of the measurement may not even be in the k candidate measurements in the tree. Therefore, the updated target estimate may not correspond to a node in the tree. Instead, we compute the node in the tree whose target estimate is closest to the actual one. We can use Bhattacharyya distance [7] to find the closest target estimate in the tree. The corresponding node then becomes the new root node of the tree. The optimal policy starting at that node is executed, iteratively. If the Bhattacharyya distance of the closest node is too large, we still can rebuild the whole minimax tree with the current target estimate as the root node.

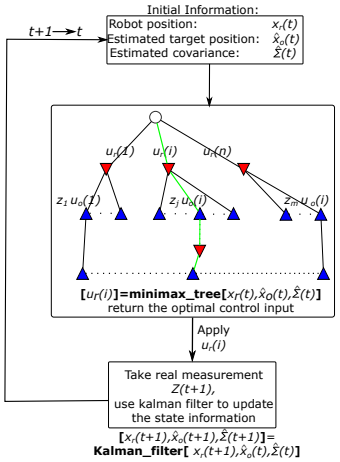


Fig. 7. Online execution of the minimax tree. At each time step, the robot executes the first control action given by the tree and obtains a measurement of the target. If the measurement $z(t+1)$ is close to one of the existing nodes, then that node is termed as the new root node. The tree can then extended to have a depth of $(k+1)$. If the new measurement $z(t+1)$ is not close to any nodes, then the tree can be rebuilt with the current estimate as the root node.

When rebuilding the tree, solving the LMI given in Conditions (C4) and (C5) may be prohibitively slow. Instead, we can restrict the comparison to just a pair of nodes which takes significantly less time.

E. Trajectory Optimization

One of the assumptions we make is that the robot and the target have a set of finite, discrete control inputs. The size of the tree (without pruning) is exponential in the number of control actions. Therefore, the minimax tree approach is reasonable when the number of control input is limited. However, in practice, the set of control inputs available to the robot could be large, potentially infinite. In such cases, the proposed tree search algorithm will not scale.

In such cases, we can use the minimax search approach presented along with a trajectory optimization algorithm (e.g., [31]). Trajectory optimization methods take as input an initial trajectory and then refine it so as to improve the quality of the trajectory. We can use the path produced by the minimax search as an initial path that is furthere refined by trajectory optimization. For scalability, we can choose a small set of control inputs for the robot and the target to build the minimax search tree. Then, we can use a trajectory optimization method, e.g., Iterative Linear Quadratic Gaussian (iLQG) [31], that is not restricted to the smaller set of control inputs to refine the trajectory. In an online setting, we will execute only the first control input in the refined trajectory obtained. The process is then repeated after every time step as described in the previous subsection.

Different initial trajectories given to trajectory optimization lead to different output trajectories especially in environments

with obstacles. The minimax search generates a better initial trajectory. Figure 8 provides an example that different initial trajectories will lead to different results. In Figure 8, two different initial collision-free trajectories are computed considering limited control inputs (up, down, left, right). In this example, to simplify the comparison, we consider the target's path is fixed. We see the effect of the initial trajectory given to the optimazation routine.

The trajectory optimization algorithm is allowed to pick control inputs for the robot and the target within a unit ball at every time step. That is, $||u_r(t)||_2 \leq 1$ and $||u_o(t)||_2 \leq 1$. However, any other suitable control input space can be used. Given the initial trajectory, we iteratively sample control inputs within a unit ball of the position of the robot and the target given in the initial trajectory, for each time step. The objective function for trajectory optimization is still Equation 5 (trace of the covariance in the final step). We alternate between trajectory optimization for the robot and the target. That is, we first optimize the trajectory for the robot assuming that the target's trajectory is fixed. We then fix the optimized robot's trajectory, and find the trajectory for the target. This process is then repeated for a fixed number of iterations or until a time deadline is met.

We follow similar steps in the previous subsection when we run the minimax with trajectory optimization algorithms online. The robot executes the first control action along the optimal trajectory. Then, the robot obtains a measurement and repeats the process if the measurement does not correspond to the worst-case measurement.

With an initial path generated by minimax, the trajectory optimization approach can improve the overall performance and compensate for the disadvantage of minimax that we only consider limited control inputs. We can use coarser discretization to get a faster trajectory first and then refine it with trajectory optimization. In this way, we can balance the trade-off between the accuracy and speed of the algorithm. We present more results in the simulation section.

V. Simulations

We carry out four types of evaluations via simulations. First, to show the efficacy of the minimax tree search in adversarial target tracking, we present qualitative and quantitative results. These results show the advantage of applying the minimax over three baseline approaches. Second, we investigate the computational savings due to our algorithm by comparing the number of nodes in the pruned minimax tree and the full enumeration tree. Then, we study the effect of varying the ϵ_1 and ϵ_2 parameters on the number of nodes. Finally, we use the control policy given by our algorithm and execute it by drawing actual measurements from a random distribution. This represents a realistic scenario where the measurements are not necessarily adversarial. We demonstrate how our strategy can be used in such a case, and compare the average-case with the worst-case performance.

A. Comparisons with Baseline Approaches

We compare the performance of the minimax tree search with three baseline approaches. We use two environments as shown in Figure 11. We assume that the robot can move faster than the target. The robot has the following control inputs, $\mathcal{U}_r$, to choose from,

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}, \begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}, \begin{bmatrix}-\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}, \begin{bmatrix}-\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}.$$

The target moves slower and only has the following four control inputs,

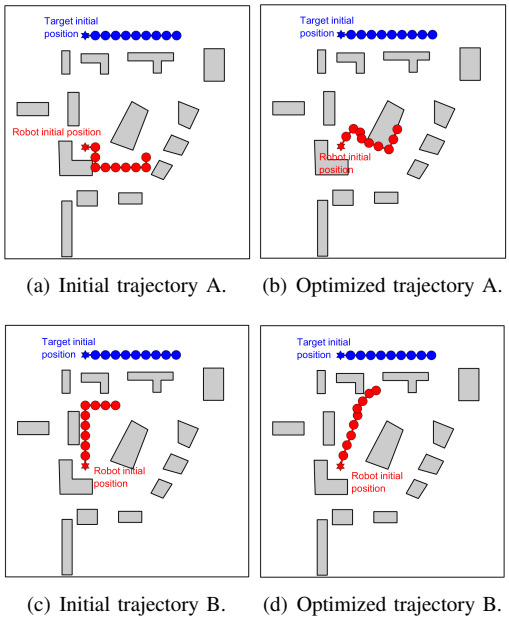
$$\mathcal{U}_o = \left\{\begin{bmatrix}0.5\\0\end{bmatrix}, \begin{bmatrix}0\\0.5\end{bmatrix}, \begin{bmatrix}-0.5\\0\end{bmatrix}, \begin{bmatrix}0\\-0.5\end{bmatrix}\right\}$$

Even though the target moves slower than the robot, it can fly across the obstacles since we model it as an aerial robot. The robot, on the other hand, moves on the ground plane and therefore cannot pass through obstacles.

We use three baselines: a greedy strategy, Dynamic Programming (DP) [2], [23], and tree search without adversarial nodes. The greedy strategy is to choose a control input that takes the robot closest to the mean of the target's estimate. The DP strategy maximizes $J(T)$ which is given by the following recurrence:

$$J(t) = \min_{u_r(t)} \left(\text{tr}(\hat{\Sigma}_t) + J(t-1)\right). \tag{15}$$

This objective function is the cumulative sum of the traces whereas in minimax we only optimize the final trace at the end of the planning horizon. The third baseline strategy is the same as minimax but without the MAX levels. That is, we will no longer consider the adversarial motion of the target (instead plan considering that the target is stationary).



(a) Initial trajectory A. (b) Optimized trajectory A.

(c) Initial trajectory B. (d) Optimized trajectory B.

Fig. 8. A robot moves in a 2-D environment with obstacles. (a) and (c) are the initial collision-free trajectory computed by considering limited control input (up, down, left, right). (b) and (d) are the output of the trajectory optimization for ten times of iteration steps. The initial positions are marked in the hexagram. We set the initial trace of the covariance $tr(\Sigma_0) = 20$. With initial trajectory A, the trace of the covariance is 2.09 at the final step after applying trajectory optimization. With initial trajectory B, the trace of the covariance is 1.59 at the final step.

TABLE II
QUANTITATIVE EXAMPLES IN DIFFERENT ENVIRONMENTS, EACH TRIAL STARTS WITH DIFFERENT INITIAL POSITIONS.

	Average final trace of the covariance matrix after 50 steps			
Environment	**Greedy**	**DP**	**Tree search without adversarial**	**Minimax tree search**
A	1.473	0.931	0.986	0.684
B	1.856	1.571	1.615	1.346

The baseline strategies consider expected measurements when planning over the horizon. The evaluation is always done online for each of the four strategies: at each time step, we execute the action given by the planner, take an actual measurement, update the target's estimate, and replan.

In general, minimax is a non-myopic planning algorithm and can plan on a longer horizon. If the target is actively escaping from the robot, minimax can predict the adversarial moves of the target. Thus minimax can better tack the target than DP and tree search without adversarial planning. This is reflected in the two qualitative examples shown in Figures 9 and 10.

We also show how the trace of the covariance matrix $\text{tr}(\Sigma_t)$ evolves over time. The initial covariance matrices are $\Sigma_0 = I$. From Figure 9 and Figure 10, the greedy algorithm performs poorly since it is myopic and can be easily blocked by the obstacles. The DP has better tracking results than the greedy. However, in these two qualitative examples, the target chooses the optimal path to escape. The minimax algorithm can better predict the target's movement than a tree search without adversarial planning and DP. In Figure 10(b), even using the non-myopic DP, the robot was stuck, giving the planning horizon of five.

We also performed quantitative comparisons for the scenarios. We compare the tracking performance of the four approaches. For both environments, we vary the starting position of the robot (the target always start in the center). Table II shows the average of the trace of the covariance matrix after 50 time steps. Not surprisingly, all strategies perform better in environment A since it is more open than B. The greedy strategy performs the worst. The minimax tree search outperforms the other baseline.

B. Comparing the Number of Nodes

In this section, the models used in the simulation is as follows. The robot follows a linear motion model and can choose from four actions at each time step, $X_r(t+1) = X_r(t) + u_r(t)$,

$$\mathcal{U}_r = \left\{ \begin{bmatrix} 1.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1.0 \end{bmatrix}, \begin{bmatrix} -1.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1.0 \end{bmatrix} \right\} \tag{16}$$

We build the tree using five candidate measurements at each step: $z(t) = \{z_1(t), z_2(t), \cdots, z_5(t)\}$. The five values are randomly generated by drawing from a Gaussian distribution. The rest of the parameters are $C_t = I, H = I, \Sigma_v = I, \delta_1 = 0.5, \delta_2 = 0.1, \mathcal{B} = 1$. We assume the target is stationary in this section.

Just like Algebraic Redundancy in [33], (C4) and (C5) can be checked using an LMI solver. However, solving for an LMI is computationally expensive. A simpler method is to only check pairs of nodes. That is, when a new node A is generated, check node A and only one of the candidate nodes one by one. This results in lesser pruning but faster checks.

In our experiment, we check the candidate nodes one by one as follows. To check (C1)–(C5) in Theorem 2 or Theorem 4, for each node in the search tree, we store its current level (depth in the search tree), current position X_r, the estimated target position $\hat{X}_o$, the covariance matrix, and a vector that stores the ancestors. We also need a list to store the nodes that are along the optimal minimax path (green path in Figure 6). For a newly generated node A, we do the following steps:

- For the nodes at the same level, find the nodes where the robot and target's estimated position are the same as node A (condition (C1)).
- For all the nodes we found, only keep the nodes along the optimal minimax path and then find their common ancestors with node A (conditions (C2) and (C3)).
- Check condition (C4) or (C5) based on the type of the common ancestor (MIN or MAX).
- If (C4) or (C5) is true, mark node A as being pruned and not explore its children nodes.

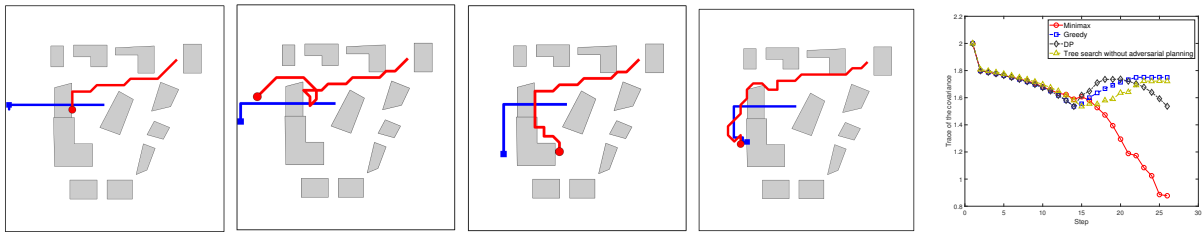
Figure 12 shows an example of a five-level minimax tree with pruning. Figure 13 shows the number of nodes in the minimax tree after pruning and the number of nodes in a full enumeration tree, respectively. We prune a node by comparing it to the nodes already explored. More nodes will be pruned if initial nodes encountered are "close" to the optimal policy. For instance, if the first set of nodes explored happen to be the optimal control law that drives the robot close to the target, then we expect the nodes encountered later will be pruned earlier in the process. To provide a fair assessment, we generate the search trees for various true positions of the target. Figure 13 shows the average and standard deviation of the number of nodes.

Figure 13 shows that our algorithm prunes orders of magnitudes of nodes from the full enumeration tree. For a tree with depth 13, there are 8.08×10^7 nodes in the full tree but the same optimal solution can be computed using 4.36×10^5 nodes with our pruning strategy.

C. Comparing the Sub-optimal Pruning algorithm

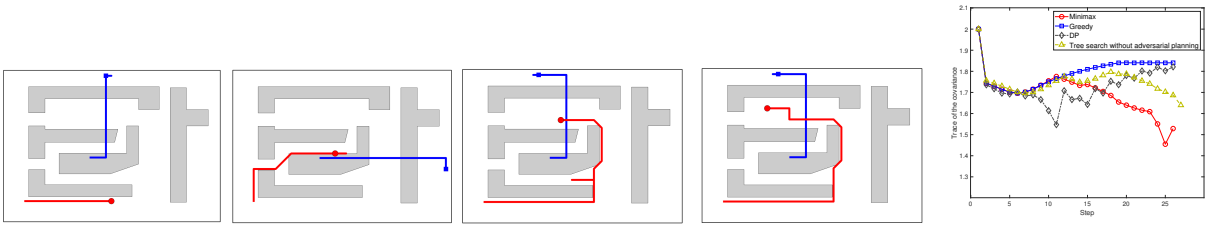
By sacrificing optimality, we can prune even more nodes. We evaluate this by varying ϵ_1 and ϵ_2 individually first, and then jointly. As shown in Figure 14, ϵ_1-alpha pruning is relatively better at reducing the size of the minimax tree. This is intuitive because ϵ_1-alpha pruning condition compares nearly every pair of nodes at the same depth. ϵ_2-algebraic redundancy pruning, on the other hand, requires more conditions to be satisfied. Nevertheless, Figure 14 shows that by sacrificing optimality, the number of nodes can be substantially reduced.

We also study the effect that varying ϵ_1 and ϵ_2 has on the optimality. Figure 15 plots the upper and lower bounds for the trace of the covariance in the optimal case ($J^*_{2k} = \text{tr}(\hat{\Sigma}^*_{2k})$) as well as in the suboptimal cases ($J^{\epsilon_1}_{2k}$ and $J^{\epsilon_2}_{2k}$). The lower bound of the optimal value J^*_{2k} is obtained by applying the Kalman Riccati equation k times to the initial covariance matrix Σ_0 assuming that the distance $d(X_r(t), X_o(t)) = 0$ for all t. This



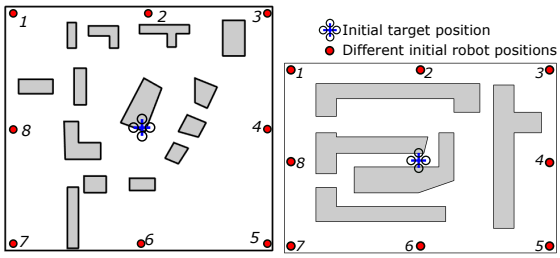
(a) Greedy (b) DP (c) Tree search without adversarial planning (d) Minimax (e) Trace of covariance matrix at each step.

Fig. 9. Qualitative example 1: (a)–(d) provide the path for the robot (red) and the target (blue), given by greedy, DP, tree search without adversarial planning, and minimax. (e) is the comparison of the updated trace of covariance matrix $\text{tr}(\Sigma_t)$ at each step. The size of the environment is a 25×25. For DP and minimax, the planning horizon for each step is 5 (The height of the minimax tree is 11).



(a) Greedy (b) DP (c) Tree search without adversarial planning (d) Minimax tree search (e) Trace of covariance matrix at each step in these four cases.

Fig. 10. Qualitative example 2: (a)-(d) provide the path for the robot (red) and the target (blue), given by greedy, DP, tree search without adversarial planning, and minimax. (e) is the comparison of the updated trace of covariance matrix $\text{tr}(\Sigma_t)$ at each step. The size of the environment is a 21×15.



(a) Environment A (25×25). (b) Environment B (21×15).

Fig. 11. Environments used for the online simulations. Blue dots are the different initial positions for the robot.

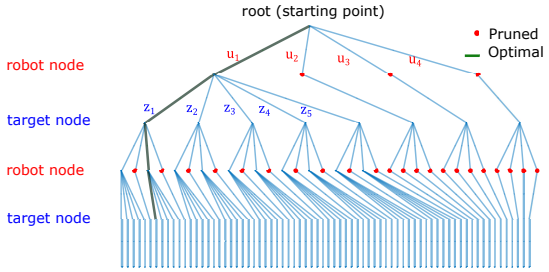


Fig. 12. A five-level minimax tree with pruning (189 nodes). Full enumeration has 505 nodes.

corresponds to the case when the variance of the noise is minimum. The upper bound is obtained by considering the worst case noise which occurs when $||X_r(t) - X_o(t)||_2 = \mathcal{B}$. The solution of a minimax search tree, J^*_{2k}, lies between the upper and lower bounds (i.e., between the two blue curves).

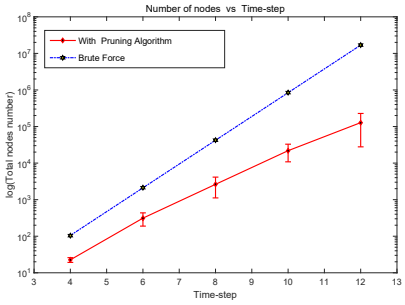


Fig. 13. Comparison of the number of total nodes generated for minimax tree. Note that the y axis is log scale.

From the upper bound of J^*_{2k}, we can plot the upper bounds for $J^{\epsilon_1}_{2k}$ and $J^{\epsilon_2}_{2k}$. The upper bound for $J^{\epsilon_1}_{2k}$ is given by Theorem 3 whereas that for $J^{\epsilon_2}_{2k}$ is given by Theorem 4. For the latter, we use the worst-case measurements to compute the recursive term. We use $\epsilon_1 = 0.1$, $\epsilon_2 = 0.2$ and $\mathcal{B} = 1$ to plot the figure. When we have $\epsilon_1 = 0.1$ and $\epsilon_2 = 0.2$ together, the bound will be the maximum of the upper bounds for $J^{\epsilon_1}_{2k}$ and $J^{\epsilon_2}_{2k}$.

Table III shows the error in tracking the target target caused by sacrificing optimality (i.e., non-zero ϵ_1 and ϵ_2). In this experiment, the robot starts at $(0, 0)$ and the target's initial position is $(1, 0)$. We run the policy for $k = 7$ time steps. We assume the robot and target can move with the same speed, $\mathcal{U}_r = \mathcal{U}_o = 1$. The initial distance between the robot and the target is 1. Table III shows the average distance (of 50 trials) after k time steps using a suboptimal policy with various

values of ϵ_1 and ϵ_2. The table shows that the average separation increases as ϵ_1 and ϵ_2 increase, as expected. We also observe that ϵ_1 has a larger effect on the tracking error as compared to ϵ_2.

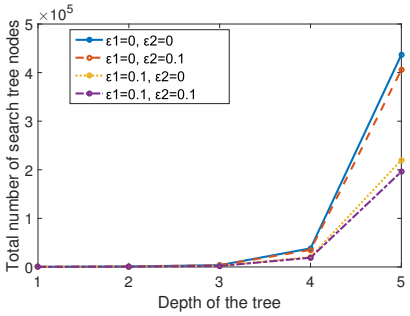


Fig. 14. Effect of the ϵ_1 and ϵ_2 relaxation parameters on the number of nodes in the search tree. The baseline case is the optimal solution with alpha pruning and algebraic redundancy with both parameters set to zero.

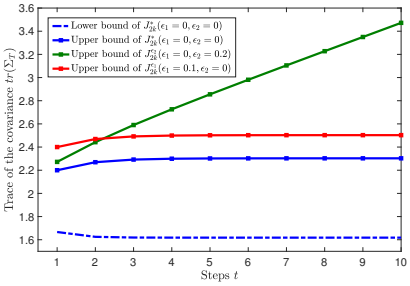


Fig. 15. Applying Thereom 3 and Thereom 4. The range of $J^*_{2k} = \mathrm{tr}(\hat{\Sigma}^*_{2k})$, $J^{\epsilon 1}_{2k}, J^{\epsilon 2}_{2k}$ with planning horizon $k = 1, 2, \cdots, 10$. Initial value $\Sigma_0 = I$, $\delta_1 = 0.5$, $\delta_2 = 0.1$, $\mathcal{B} = 1$.

TABLE III
EFFECT OF THE ϵ_1 AND ϵ_2 RELAXATION PARAMETERS ON THE TRACKING ERROR.

	$\epsilon_1 = 0$, $\epsilon_2 = 0$	$\epsilon_1 = 0$, $\epsilon_2 = 0.2$	$\epsilon_1 = 0.2$, $\epsilon_2 = 0$	$\epsilon_1 = 0.2$, $\epsilon_2 = 0.2$	$\epsilon_1 = 0.3$, $\epsilon_2 = 0.3$
Average distance in 50 times	1.00	1.36	1.54	1.70	2.26

D. Trajectory Optimization

In this subsection, we compare the performance of executing the controls found by minimax only and minimax with trajectory optimization. We use the environment in Figure 11. We use minimax to generate the initial trajectory by using only four control inputs for both the robot and the target (given in Equation 16). Then we use the same objective function in Equation 5 (trace of the covariance in the final step). The trajectory optimization is allowed to refine the trajectory subject to $||u_r(t)||_2 \leq 1$ and $||u_o(t)||_2 \leq 1$. Unlike the previous simulations, we assume the target cannot go over the obstacles to show the effect of trajectory refinement.

We use this in an online setting as described in Section IV-E. The two cases we compare, minimax only and minimax with trajectory optimization, corresponding to the algorithm that is used by the robot to choose its control inputs. In both cases, the target uses minimax with trajectory optimization (even if the robot is not). This is because in practice the target is not restricted to use a minimax tree or subject to only follow the four control inputs we use to build the tree. The goal of this experiment is to show how much improvement we can get with trajectory optimization in terms of the tracking performance.

TABLE IV
QUANTITATIVE EXAMPLES FOR MINIMAX ONLY VS MINIMAX WITH TRAJECTORY OPTIMIZATION.

	Final trace of the covariance matrix after 25 steps mean (standard deviation)	
Environment	**Minimax only**	**Minimax with optimization**
A	0.97 (0.16)	0.73 (0.13)
B	1.87 (0.29)	1.12 (0.25)

A qualitative example is shown in Figure 16. In Figure 16-(a), we see the two trajectories followed by the robot using minimax only (left) and minimax with trajectory optimization (right). We observe that the robot is able to get closer to the target, follow a smoother trajectory, as well as improve the performance in tracking. This can be seen in Figure 16-(b) which shows the improvement of the trace of the covariance.

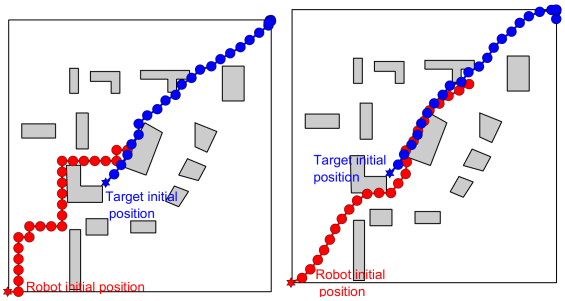
We also performed quantitative comparisons. For the two environments in Figure 11, we vary the starting position of the robot (the target always start in the center). Table IV shows the mean and the standard deviation of the trace of the covariance matrix after 25 time steps. The trajectory optimization approach can always improve the path found by the minimax tree. We observe that the improvement is larger in environment B. We suspect that this is because environment B has longer, narrower corridors where if a robot goes down an incorrect path it does not get easy opportunities to correct it. On the other hand, environment A has smaller obstacles that the robot can go around. Therefore, the trajectory optimization results in larger improvement in the more challenging environment (B).

VI. EXPERIMENTS

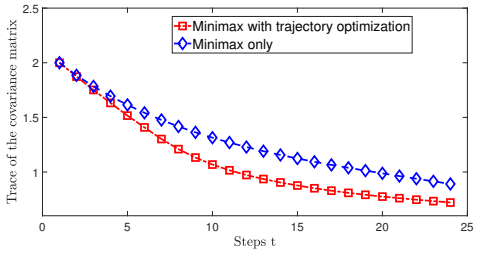
We implemented the worst-case minimax tracking algorithm using indoor and outdoor robots. We use a five-level minimax tree with a look-ahead of two steps. Our experiments show that the algorithm can be successfully implemented and executed on real hardware.[7]

For the indoor experiment, we used two Pioneer 3DX robots (Figure 17) equipped with a 2.6GHz i7-6700HQ processor and 16 GB RAM to find the optimal minimax strategy. We use MATLAB 2015b to execute the proposed algorithm and send the control inputs to the Pioneer 3DX robot through MATLAB ROS toolbox. One Pioneer acts as the target and the other acts as the tracking robot. The motion and measurement models and parameters are similar to the Gazebo simulation reported in the previous section. The measurement noise is generated using parameters $\delta_1 = 0.5$ and $\delta_2 = 0.1$ from the true position of the target robot. The robot's speed is $0.4m/s$ and the target's speed is $0.25m/s$. We use minimax with optimal pruning

[7]The video can be found at https://youtu.be/aU0x25Ak4vs.

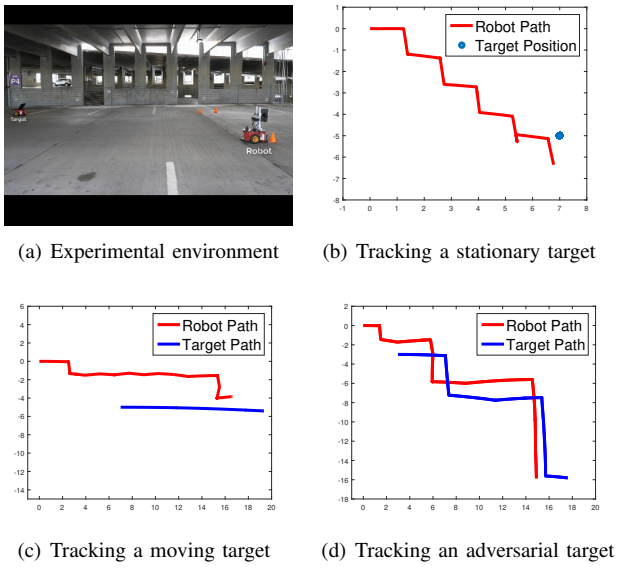


(a) Online adversarial planing: Minimax only vs minimax with trajectory optimization.



(b) The trace of the covariance at each step.

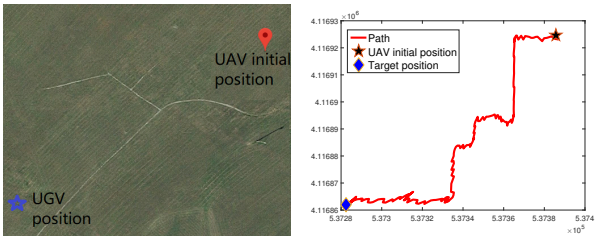
Fig. 16. Qualitative online path planning examples for minimax with trajectory optimization: (a) provides the path for the robot (red) and the target (blue), given with/without trajectory optimization. The initial positions are marked by a hexagram. (b) show the comparison of the trace of covariance matrix $\text{tr}(\Sigma_t)$ at each step. Note that this is an online execution where the robot replans at every timestep. This is why the optimized trajectory deviates from the minimax only trajectory in terms of going around the obstacle.



(a) Experimental environment (b) Tracking a stationary target

(c) Tracking a moving target (d) Tracking an adversarial target

Fig. 17. Real-world indoor tracking experiments.



(a) Experimental environment (b) UAV online tracking

Fig. 18. Real-world outdoor tracking experiment.

conditions in real-world experiments. The results presented in this section are found by minimax only, without trajectory optimization.

The robot only takes a new measurement and computes the next control input after it finishes the previous movement. In this experiment, we use a 5 level minimax tree (look ahead for two steps) to compute the strategy. It takes on an average (of 10 trials) $1.36s$ to compute the policy with a 5 level minimax tree. The computation time may vary based on the specific instance and the parameters.

We carried out three sets of experiments: (1) tracking a stationary target; (2) tracking a target that moves in a straight line; and (3) tracking a target that actively chooses adversarial control inputs to evade the tracker. Figure 17 shows the robot and the target's trajectories for the three experiments. In all cases, we see that the minimax algorithm with pruning drives the robot towards the target. Furthermore, the hardware experiments demonstrate that the minimax algorithm can be applied in real-time on actual hardware. Videos from the experiments are reported in the accompanying multimedia file.

The outdoor experiment consisted of an Unmanned Aerial Vehicle (UAV) tracking a stationary target using the minimax tree. The UAV uses a DJI Flame Wheel F450 frame and ArduPilot Mega (APM) firmware running on a Pixhawk autopilot. The on-board computer interfaces with the autopilot using the mavros package of the Robotic Operating System (ROS). The same computer used for the indoor experiment runs the minimax search algorithm and sends the waypoints to the autopilot. The experiments were conducted in a farmland near Blacksburg, VA, USA. The measurement is obtained by a noisy GPS sensor placed on the target. In order to generate the tree, the GPS noise is modeled as a zero mean Gaussian noise with constant variance ($\delta_1 = 0.5$ and $\delta_2 = 0.1$). The starting position of the UAV was about 160 meters from the target.

The UAV took off manually and then switched to the autonomous mode, where it follows the control commands given by the minimax tree. Figure 18 shows the resultant trajectory of the UAV produced by the algorithm. Similar to the indoor experiment, the UAV has four motion control input (forward, backward, left, right). The unit step of the UAV is set as 10 meters. The accompanying multimedia attachment contains a video of the outdoor experiment.

The indoor and the outdoor experiments demonstrate that the online minimax tracking algorithm along with the pruning strategy can be applied in real-time on actual hardware.

VII. Conclusion

We investigated the problem of devising closed-loop control policies for target tracking with distance-dependent measurement noise. Unlike the state-independent noise case, the value of a candidate control law in our version is a function of

the history of measurements obtained. Consequently, planning over a horizon requires taking into account all possible measurement values. We focused on minimizing worst-case uncertainty. Our solution consists of building a minimax search tree to obtain the control policy. A full enumeration tree has a size that is exponential in the number of measurements, control actions, and the planning horizon. Instead, we exploited the structural properties of Kalman filter to yield a tree with significantly less number of possible nodes without sacrificing the optimality guarantees. We also showed how two parameters, ϵ_1 and ϵ_2, can be used to yield even more computational savings at the expense of optimality. The resulting algorithm was evaluated in simulations and through real-world experiments.

One disadvantage of the generalization is that we have to discretize the set of possible future measurements. Our immediate future work is to bound the suboptimality as a function of the number of discrete samples chosen to represent the continuous set of measurements. The algebraic redundancy conditions require the states to be identical which is reasonable when operating in a discrete setting. A useful extension would be to group together nearby states, and quantify the effect of such grouping, so that we can allow for even more pruning and/or extend to the continuous regime. Another avenue of future work focuses on extending these results to multi-robot, multi-target scenarios. Our prior work [27], [29] has shown a greedy assignment of robot trajectories to targets yield provably approximate solutions for one-step planning. We will extend this to plan over a finite horizon using the results presented in this paper.

APPENDIX A
PROOF OF LEMMA 1

Proof. For nodes A and B at the same level k with $S^A \succeq S^B$, and $\Sigma_t^A \succeq \Sigma_t^B$. our goal is to prove that $\rho_t(\Sigma_t^A) \succeq \rho_t(\Sigma_t^B)$.

Applying the Riccati equation we have:

$$\begin{aligned}&\rho_t(\Sigma_t^A) - \rho_t(\Sigma_t^B)\\ =&C_t\Sigma_t^A C_t^T - C_t\Sigma_t^A H^T\left(H\Sigma_t^A H^T + S^A\right)^{-1} H\Sigma_t^A C_t^T\\ &- C_t\Sigma_t^B C_t^T + C_t\Sigma_t^B H^T\left(H\Sigma_t^B H^T + S^B\right)^{-1} H\Sigma_t^B C_t^T.\end{aligned} \tag{17}$$

Define,

$$\begin{aligned}K(\Sigma_t) &\triangleq -C_t\Sigma_t H^T(H\Sigma_t H^T + S)^{-1},\\ F(\Sigma_t) &\triangleq C_t - (C_t\Sigma_t H^T)(H\Sigma_t H^T + S)^{-1}H.\end{aligned}$$

Note that,

$$\begin{aligned}F(\Sigma_t) &= C_t + K(\Sigma_t)H,\\ K(\Sigma_t)(C_t\Sigma_t H^T)^T &= -K(\Sigma_t)(H\Sigma_t H^T + S)K^T(\Sigma_t).\end{aligned}$$

Thus,

$$\begin{aligned}\rho_t(\Sigma_t^A) - \rho_t(\Sigma_t^B) =&C_t\Sigma_t^A C_t^T + K(\Sigma_t^A)H\Sigma_t^A C_t^T - C_t\Sigma_t^B C_t^T\\ &- K(\Sigma_t^B)H\Sigma_t^B C_t^T.\end{aligned} \tag{18}$$

Then, we have,

$$\begin{aligned}&\rho_t(\Sigma_t^A) - \rho_t(\Sigma_t^B) - F(\Sigma_t^A)(\Sigma_t^A - \Sigma_t^B)F(\Sigma_t^A)^T\\ =&C_t\Sigma_t^A C_t^T + K(\Sigma_t^A)H\Sigma_t^A C_t^T\\ &- [C_t\Sigma_t^B C_t^T + K(\Sigma_t^B)H\Sigma_t^B C_t^T]\\ &- [C_t + K(\Sigma_t^A)H](\Sigma_t^A - \Sigma_t^B)[C_t + K(\Sigma_t^A)H]^T\\ =&K(\Sigma_t^A)(C_t\Sigma_t^A H)^T - K(\Sigma_t^B)C_t\Sigma_t^B H^T\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)C_t^T - C_t(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A)\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A)\\ =&K(\Sigma_t^A)(C_t\Sigma_t^A H)^T - K(\Sigma_t^B)C_t\Sigma_t^B H^T\\ &- K(\Sigma_t^A)[H\Sigma_t^A C_t^T - H\Sigma_t^B C_t^T]\\ &- [C_t\Sigma_t^A H^T - C_t\Sigma_t^B H^T]K^T(\Sigma_t^A)\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A)\\ =&K(\Sigma_t^A)(C_t\Sigma_t^A H)^T - K(\Sigma_t^B)C_t\Sigma_t^B H^T\\ &- K(\Sigma_t^A)H\Sigma_t^A C_t^T + K(\Sigma_t^A)H\Sigma_t^B C_t^T\\ &- C_t\Sigma_t^A H^T K^T(\Sigma_t^A) + C_t\Sigma_t^B H^T K^T(\Sigma_t^A)\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A),\end{aligned} \tag{19}$$

note the following matrix is symmetric,

$$K(\Sigma_t)(C_t\Sigma_t H^T)^T = -K(\Sigma_t)(H\Sigma_t H^T + S)K^T(\Sigma_t),$$

we have,

$$K(\Sigma_t^A)(C_t\Sigma_t^A H)^T = C_t\Sigma_t^A H^T K^T(\Sigma_t^A).$$

The first and the fifth term in (19) can be canceled,

$$\begin{aligned}=&- K(\Sigma_t^B)C_t\Sigma_t^B H^T - K(\Sigma_t^A)H\Sigma_t^A C_t^T\\ &+ K(\Sigma_t^A)H\Sigma_t^B C_t^T\\ &+ C_t\Sigma_t^B H^T K^T(\Sigma_t^A)\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A)\\ =&- K(\Sigma_t^B)C_t\Sigma_t^B H^T\\ &+ K(\Sigma_t^A)(H\Sigma_t^A H^T + S^A)K^T(\Sigma_t)\\ &+ K(\Sigma_t^A)H\Sigma_t^B C_t^T + C_t\Sigma_t^B H^T K^T(\Sigma_t^A)\\ &- K(\Sigma_t^A)H(\Sigma_t^A - \Sigma_t^B)H^T K^T(\Sigma_t^A),\end{aligned}$$

combine the second and the last term,

$$\begin{aligned}=&- K(\Sigma_t^B)(C_t\Sigma_t^B H^T) + K(\Sigma_t^A)(C_t\Sigma_t^B H^T)^T\\ &+ (C_t\Sigma_t^B H^T)K^T(\Sigma_t^A)\\ &+ K(\Sigma_t^A)(H\Sigma_t^B H^T + S^A)K^T(\Sigma_t^A),\end{aligned}$$

note that,

$$(C_t\Sigma_t H^T)^T = (H\Sigma_t H^T + S)K^T(\Sigma_t)$$

$$\begin{aligned}=&\quad K(\Sigma_t^B)(H\Sigma_t^B H^T + S^B)K^T(\Sigma_t^B)\\ &- K(\Sigma_t^A)(H\Sigma_t^B H^T + S^B)K^T(\Sigma_t^B)\\ &- K(\Sigma_t^B)(H\Sigma_t^B H^T + S^B)K^T(\Sigma_t^A)\\ &+ K(\Sigma_t^A)(H\Sigma_t^B H^T + S^A)K^T(\Sigma_t^A)\\ =&(K(\Sigma_t^B) - K(\Sigma_t^A))(H\Sigma_t^B H^T + S^B)(K(\Sigma_t^B) - K^T(\Sigma_t^A))^T\\ &+ K(\Sigma_t^A)(S^A - S^B)K^T(\Sigma_t^A).\end{aligned} \tag{20}$$

We have,

$$\begin{aligned}
&\rho_t(\Sigma_t^A) - \rho_t(\Sigma_t^B) \\
=&F(\Sigma_t^A)(\Sigma_t^A - \Sigma_t^B)F^T(\Sigma_t^A) + K(\Sigma_t^A)(S^A - S^B)K^T(\Sigma_t^A) \\
&+ (K(\Sigma_t^B) - K(\Sigma_t^A))(H\Sigma_t^B H^T + S^B)\cdot \\
&\quad (K(\Sigma_t^B) - K(\Sigma_t^A))^T \\
=&C_t H(\Sigma_t^A - \Sigma_t^B)H^T C_t^T \\
&+ K(\Sigma_t^A)\left((H\Sigma_t^A H^T + S^A) - (H\Sigma_t^B H^T + S^B)\right)K^T(\Sigma_t^A) \\
&+ (K(\Sigma_t^B) - K(\Sigma_t^A))(H\Sigma_t^B H^T + S^B)\cdot \\
&\quad (K(\Sigma_t^B) - K(\Sigma_t^A))^T
\end{aligned} \tag{21}$$

since $S^A \succeq S^B$, we have $H\Sigma_t^A H^T + S^A \succeq H\Sigma_t^B H^T + S^B$, and $\Sigma_t^A \succeq \Sigma_t^B$. Thus,

$$\rho(\Sigma_t^A) - \rho(\Sigma_t^B) \succeq 0 \tag{22}$$

□

Appendix B
Proof of Theorem 2

Proof. We first prove a special case when $\mathcal{H}$ consists of only one node, B. Then, the condition (C4) is, $\Sigma_t^A \succeq \Sigma_t^B$ and $S_t^A \succeq S_t^B + MaI_{2\times 2}$. From these two conditions, we have:

$$H\Sigma_t^A H^T \succeq H\Sigma_t^B H^T + M \cdot aI$$

where, $a = \delta_1^2 + \delta_2^2\mathcal{C}$, and $M = T - t$. The goal is to prove the following:

$$\text{tr}(\Sigma_{t+M}^A) \geq \text{tr}(\Sigma_{t+M}^B). \tag{23}$$

We will prove this by induction. We start with the base case.

Base step: Show that Equation 23 holds for $M = 1$.

When $M = 1$, $H\Sigma_t^A H^T \succeq H\Sigma_t^B H^T + aI$. From the Kalman filter Riccati map,

$$\begin{aligned}
&\Sigma_{t+1}^A = \rho_i(\Sigma_t^A) \\
=&C_t\Sigma_t^A C_t^T - C_t\Sigma_t^A H^T \cdot \\
&\left(H\Sigma_t^A H^T + \Sigma_{w_i}\left(x_r(t), \hat{x}_o^A(t)\right)\right)^{-1} H\Sigma_t^A C_t^T + \Sigma_v
\end{aligned} \tag{24}$$

Applying Lemma 1, we get,

$$\begin{aligned}
&\Sigma_{t+1}^A \\
\succeq&C_t\Sigma_t^B C^T - C_t\Sigma_t^B H^T \cdot \\
&\left(H\Sigma_t^B H^T + \Sigma_{w_i}\left(x_r(t), \hat{x}_o^B(t)\right)\right)^{-1} H\Sigma_t^B C_t^T + \Sigma_v \\
=&\Sigma_{t+1}^B
\end{aligned} \tag{25}$$

Inductive step: Show that if Equation 23 holds for $M = k$, then it also holds for $M = k+1$. This can be done as follows:

$$H\Sigma_t^A H^T \succeq H\Sigma_t^B H^T + (k+1) \cdot aI$$

Rewrite the equation as:

$$H\Sigma_t^A H^T \succeq [H\Sigma_t^B H^T + aI] + k \cdot aI.$$

Since $M = k$ holds for some k. Let $H\Sigma_t^{B'} H^T = H\Sigma_t^B H^T + aI$, based on the condition of $M = k$ we have,

$$\Sigma_{t+k}^A \succeq \Sigma_{t+k}^{B'}$$

that is,

$$\Sigma_{t+k}^A \succeq \Sigma_{t+k}^B + a \cdot (H^T H)^{-1}.$$

Similar to the base step:

$$\Sigma_{t+k+1}^A \succeq \Sigma_{t+k+1}^B$$

Thereby showing that indeed $M = k + 1$ holds.

Therefore, by mathematical induction, Equation 23 holds for all integers M. Next, we extend this result from $|\mathcal{H}| = 1$ to N comparable nodes in $\mathcal{H}$. Our goal is to prove at the terminal time T:

$$\begin{aligned}
&\text{tr}\left(H\left(\Sigma_t^A\right)H^T\right) \geq \\
&\qquad \sum_{i=1}^{i=N} \alpha_i \text{tr}\left(\left[H\Sigma_t^i H^T + K \cdot aI\right]\right)
\end{aligned} \tag{26}$$

Without loss of generality, we assume Σ^B is the minimum covariance matrix among $\Sigma_t^1, \Sigma_t^2, ..., \Sigma_t^N$. We have:

$$\begin{aligned}
&\sum_{i=1}^{i=N} \alpha_i \left(\left[H\Sigma_t^i H^T + N \cdot aI\right]\right) \\
&\qquad \succeq \sum_{i=1}^{i=N} \alpha_i \left(\left[H\Sigma_t^B H^T + N \cdot aI\right]\right) \\
&\qquad = H\Sigma_t^B H^T + N \cdot aI
\end{aligned} \tag{27}$$

So,

$$\text{tr}\left(H\left(\Sigma_t^A\right)H^T\right) \geq \text{tr}\left(H\Sigma_t^B H^T + K \cdot aI\right) \tag{28}$$

This conditions says that the trace of the covariance at node A is greater than the trace of covariance of any successor of node B for a tree of depth at most K. Therefore, node A will not be part of the optimal policy and can be pruned without affecting optimality of the minimax tree. □

Appendix C
Proof of Theorem 4

Proof. For some level i, suppose that we prune a node on the optimal policy. We have,

$$\text{tr}(H\left(\Sigma_{2i}^{\epsilon_2}\right)H^T) \leq \text{tr}(H\left(\Sigma_{2i}^* + \epsilon_2 I\right)H^T),$$

we apply the following two proprieties from [33] (Lemma 1 and Theorem 3 in [33]), $\forall \Sigma, Q \in \mathbb{R}^{n\times n}$ and $\epsilon \geq 0$:

$$\rho_{2i}(\Sigma + \epsilon Q) \preceq \rho_{2i}(\Sigma) + F_i(\Sigma)QF_i^T(\Sigma)\epsilon \tag{29}$$

$$\begin{aligned}
&\frac{d\Phi_{2k}(\Sigma + \epsilon_2 Q)}{d\epsilon_2} = \\
&\left[\prod_{i=k-1}^{0}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)Q\prod_{i=0}^{k-1}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)^T\right]\epsilon_2,
\end{aligned} \tag{30}$$

from equation (29) (30), we have,

$$\begin{aligned}
&\Phi_{2k}(\Sigma + \epsilon_2 Q) \\
=&\Phi_{2k}(\Sigma) + o(\epsilon_2) + \\
&\left[\prod_{i=k-1}^{0}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)Q\prod_{i=0}^{k-1}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)^T\right]\epsilon_2 \\
\preceq&\Phi_{2k}(\Sigma) + \\
&\left[\prod_{i=k-1}^{0}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)Q\prod_{i=0}^{k-1}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)^T\right]\epsilon_2
\end{aligned}$$

where, $o(\epsilon_2)$ satisfies $o(\epsilon_2)/\epsilon_2 \to 0$ as $\epsilon_2 \to 0$.

Let $\left\{\hat{\Sigma}_i^*\right\}_{i=1}^k$ be the series of covariance matrices along the optimal minimax trajectory. Suppose that the sequence of covariace matrices along the optimal trajectory returned by ϵ_2–algebraic redundancy pruning algorithm is $\left\{\hat{\Sigma}_i^{\epsilon_2}\right\}_{i=1}^k$. We get,

$$\hat{\Sigma}_i^{\epsilon_2} \preceq \hat{\Sigma}_i^* + \epsilon_2 I, \quad \forall i = 1, 2, \ldots, k$$

Consider the worst case, the ϵ_2 pruning condition was used at all the remaining $2k$ levels of the minimax tree. It will be at most be pruned by k times. Thus, added all the bounded value from 1 to k. we obtain the desired bound:

$$\begin{aligned}
&0 \leq J_{2k}^{\epsilon_2} - J_{2k}^* = \mathrm{tr}(\hat{\Sigma}_k^{\epsilon_2}) - \mathrm{tr}(\hat{\Sigma}_k^*) \\
&\leq \mathrm{tr}\left\{\sum_{j=1}^{k}\left[\prod_{i=j-1}^{0}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)\prod_{i=0}^{j-1}\left(F_i(\Sigma)\Phi_{2i}(\Sigma)\right)^T\right]\epsilon_2\right\} \\
&= B^{\epsilon_2}
\end{aligned}$$

□

REFERENCES

[1] Robotics: Science and systems workshop in adversarial robotics. http://hcr.mines.edu/2018-rss-workshop/. Accessed: June 30, 2018.

[2] James Arnold, SW Shaw, and HENRI Pasternack. Efficient target tracking using dynamic programming. *IEEE transactions on Aerospace and Electronic Systems*, 29(1):44–56, 1993.

[3] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Information acquisition with sensing robots: Algorithms and error bounds. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6447–6454. IEEE, 2014.

[4] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[5] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[6] Gines Benet, Francisco Blanes, José E Simó, and Pascual Pérez. Using infrared sensors for distance measurement in mobile robots. *Robotics and autonomous systems*, 40(4):255–266, 2002.

[7] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distribution. *Bull. Calcutta Math. Soc*, 1943.

[8] M Boutayeb, H Rafaralahy, and M Darouach. Convergence analysis of the extended kalman filter used as an observer for nonlinear deterministic discrete-time systems. *IEEE transactions on automatic control*, 42(4):581–586, 1997.

[9] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299, 2011.

[10] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[11] Dongbing Gu. A game theory approach to target tracking in sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):2–13, 2011.

[12] Baoqi Huang, Lihua Xie, and Zai Yang. Tdoa-based source localization with distance-dependent noises. *IEEE Transactions on Wireless Communications*, 14(1):468–480, 2015.

[13] Volkan Isler, Narges Noori, Patrick Plonski, Alessandro Renzaglia, Pratap Tokekar, and Joshua Vander Hook. Finding and tracking targets in the wild: Algorithms and field deployments. In *International Symposium on Safety, Security, and Rescue Robotics*, 2015. to appear.

[14] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Algorithmic foundations of robotics IX*, pages 71–87. Springer, 2010.

[15] Nikhil Karnad and Volkan Isler. Modeling human motion patterns for multi-robot planning. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3161–3166. IEEE, 2012.

[16] Andreas Kolling and Stefano Carpin. Pursuit-evasion on trees by robot teams. *IEEE Transactions on Robotics*, 26(1):32–47, 2010.

[17] Panqanamala Ramana Kumar and Pravin Varaiya. *Stochastic systems: Estimation, identification, and adaptive control*, volume 986. Prentice Hall Englewood Cliffs, NJ, 1986.

[18] Frank L Lewis and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 1995.

[19] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1333–1364, 2003.

[20] Michael Montemerlo, Joelle Pineau, Nicholas Roy, Sebastian Thrun, and Vandi Verma. Experiences with a mobile robotic guide for the elderly. In *AAAI/IAAI*, pages 587–592, 2002.

[21] Zhirong Qiu, Shuai Liu, and Lihua Xie. Distributed constrained consensus with distance-dependent measurement noises. *IFAC-PapersOnLine*, 49(22):234–239, 2016.

[22] Alberto Quattrini Li, Raffaele Fioratto, Francesco Amigoni, and Volkan Isler. A search-based approach to solve pursuit-evasion games with limited visibility in polygonal environments. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1693–1701, 2018.

[23] Steven AP Quintero, Francesco Papi, Daniel J Klein, Luigi Chisci, and Joao P Hespanha. Optimal uav coordination for target tracking using dynamic programming. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4541–4546. IEEE, 2010.

[24] BSY Rao, Hugh F Durrant-Whyte, and JA Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *The International Journal of Robotics Research*, 12(1):20–44, 1993.

[25] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 2009.

[26] Wei Sun, Yunpeng Pan, Jaein Lim, Evangelos A Theodorou, and Panagiotis Tsiotras. Min-max differential dynamic programming: Continuous and discrete time formulations. *Journal of Guidance, Control, and Dynamics*, 41(12):2568–2580, 2018.

[27] Yoonchang Sung, Ashish Kumar Budhiraja, Ryan Williams, and Pratap Tokekar. Distributed simultaneous action and target assignment for multi-robot multi-target tracking. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, 2018.

[28] Pratap Tokekar, Elliot Branson, Joshua Vander Hook, and Volkan Isler. Tracking aquatic invaders: Autonomous robots for invasive fish. *IEEE Robotics and Automation Magazine*, 2013.

[29] Pratap Tokekar, Volkan Isler, and Antonio Franchi. Multi-target visual tracking with aerial robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[30] Pratap Tokekar, Joshua Vander Hook, and Volkan Isler. Active target localization for bearing based robotic telemetry. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[31] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[32] Josh Vander Hook and Volkan Isler. Pursuit and evasion with uncertain bearing measurements. In *26th Canadian Conference on Computational Geometry, CCCG 2014*. Canadian Conference on Computational Geometry, 2014.

[33] Michael P. Vitus, Wei Zhang, Alessandro Abate, Jianghai Hu, and Claire J. Tomlin. On efficient sensor scheduling for linear dynamical systems. *Automatica*, 48(10):2482–2493, October 2012.

[34] W Willman. Formal solutions for a class of stochastic pursuit-evasion games. *IEEE Transactions on Automatic Control*, 14(5):504–509, 1969.

[35] Shuang Wu, Xiaoqiang Ren, Yiguang Hong, and Ling Shi. Max-min fair sensor scheduling: Game-theoretic perspective and algorithmic solution. *arXiv preprint arXiv:1902.03594*, 2019.

[36] I Yaesh and U Shaked. Game theory approach to state estimation of linear discrete-time processes and its relation to h-optimal estimation. *International Journal of Control*, 55(6):1443–1452, 1992.

[37] Ugur Zengin and Atilla Dogan. Real-time target tracking for autonomous uavs in adversarial environments: A gradient search algorithm. *IEEE Transactions on Robotics*, 23(2):294–307, 2007.

[38] Zhongshun Zhang and Pratap Tokekar. Non-myopic target tracking strategies for non-linear systems. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 5591–5596. IEEE, 2016.