Enhancing Security in Multi-Robot Systems through Co-Observation Planning on Unsecured Trajectories

Ziqi Yang and Roberto Tron

***Abstract*—This paper addresses the critical issue of security in multi-robot systems (MRS) when facing threats from adversaries attempting to compromise the robots' control. Such compromises could lead to unauthorized access to forbidden areas, potentially causing harm or disrupting the mission. To tackle this problem, we propose a novel multi-robot planning algorithm that leverages mutual observations between robots as an additional security measure. Our approach guarantees that even in the face of adversarial movement by attackers, compromised robots cannot reach forbidden regions without missing scheduled co-observations. We achieve this by introducing *reachability* constraints into continuous trajectory planning algorithm. These constraints ensure an empty intersection between the sets of locations that the agents could potentially reach and the forbidden regions. rtron The reachability constraint is implemented with ellipsoidal over-approximation; this allows to quickly check intersection and compute gradients. Furthermore, to enhance system resilience and feasibility of the planning problem, we introduce the concept of *sub-teams*, where multiple robots form cohesive units along each route, replacing individual robot assignments. This redundancy enables robots to deviate from their assigned sub-teams and join others to perform cross-trajectory co-observations, thereby securing multiple trajectories simultaneously. To efficiently plan cross-trajectory co-observations that maintain security against plan-deviation attacks, we formulate the multi-flow problem on unsecured MRS trajectories. rtron This second part fixes the feasiblity problems of the first one, and uses the same reacheability constraints to build inter-team trajectories. We demonstrate the effectiveness and robustness of our proposed algorithm, which significantly strengthens the security of multi-robot systems in the face of adversarial threats.**

I. Introduction

Multi-robot systems (MRS) have witnessed increasing adoption in diverse fields, such as warehouse organization, surveillance, forest fire monitoring, and precision agriculture, owing to their capacity for complex task execution through cooperation and coordination [1], [2]. The use of MRS offers numerous advantages, but it also introduces new cybersecurity challenges, including unauthorized access, malicious attacks, and data manipulation [3]. Given their distributed nature and reliance on network communication, MRS are particularly vulnerable to cyber threats. In this paper, we address the specific scenario where attackers compromise robots and direct them to restricted regions, potentially leading to the exploitation of confidential information, physical property damages, or even human injuries [4]. We refer to these restricted regions as *forbidden regions*, which might contain security-sensitive equipment or human workers.

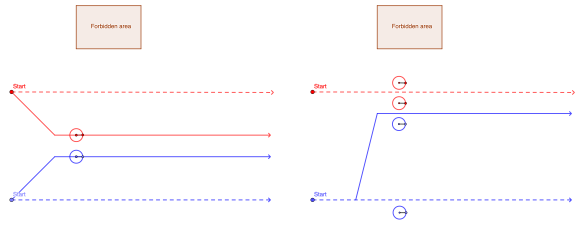
These deliberate deviations, termed *plan-deviation attacks*, have been previously identified and tackled in the literature [5] zyang add most recent articles. The work of [5] provide exact solution for grid-world configuration. Using the onboard sensing capabilities, robots can perform inter-robot observations as an additional security measure. Each robot' self-report includes not only its own status, but also additional observations of other robots' status. For example, robot i's self-report may include its observation of robot j, "i report its arrival at v and i observes j at location w". Similarly, robot j also reports its observations of robot i to the CE. This solution provides paths with an observation schedule such that any attempts by a compromised robot to violate the safety constraints would necessarily break the observation plan and be detected.

When transforming the solution to continuous configuration spaces, additional analysis is needed on the reachability of robots in order for the security of co-observation schedule to hold. More broadly, the problem of solving a path optimization problem with constraints based on the sets of locations that the agents could *potentially* reach, which we call *reachability regions*, has not received attention in the literature. This constraint is formulated using an ellipsoidal bound of the reachability region, and the idea of using an ellipsoidal bound to limit the search space is inspired by the *heuristic sampling domain* introduced by [6] in the context of the RRT* path planning algorithm.

We formulate a way to enforce an empty intersection between forbidden regions and ellipsoidal reachability region while optimizing the trajectory, such that if an attacker takes control of the robots, they cannot perform an undetected attack without breaking the co-observation schedule. The idea is inspired by the *heuristic sampling domain* introduced by [6] in the context of the RRT* path planning algorithm; in that case, an ellipsoidal bound is used to limit the search space with the initial and goal states fixed. In our case, we use a similar bound to optimize the location of the two states to exclude the forbidden region from the ellipsoidal region. We propose a mathematical formulation of reachability regions that is compatible with the solver from zyang cite ADMM as a spatio-temporal constraint.

A preliminary version of the paper was presented at conferences zyang cite CDC IROS. This paper extend the previous works by fixing two main challenges identified in [**?**] zyang previous works. Firstly, finding a co-observation and reachability secured plan may not always be feasible when robots are separated from others by obstacles or forbidden regions, or are located far away from each other. In such case, it is impossible for other robots to get close enough to establish co-observation schedules or find a reachability-secured path. Agent 3 in the Figure zyang ref fig:ReachabilitySimulation is a good example, as it required additional security checkpoint to create secured reachability areas. Secondly, security requirements may

This project is supported by the National Science Foundation grant "CPS: Medium: Collaborative Research: Multiagent Physical Cognition and Control Synthesis Against Cyber Attacks" (Award number 1932162).



(a) Co-observation (b) Cross-trajectory co-observation

Fig. 1: 1a Limited by the co-observation requirement, both red and blue robot follow the co-observation secured routes (solid lines) and abandon the optimal ones (dashed line). 1b Through cross-trajectory co-observations, the blue team sends one robot to follow the red team (solid blue line) and performs co-observation while having the rest of the robots following the optimal trajectory.

come at the cost of overall system performance, as illustrated by the comparison of Figure zyang ref fig:trajectories-more-constraint and zyang ref fig:ReachabilitySimulation where, after the introduction of security constraints, the top left corner is never explored by any robots. This is particularly concerning given that system performance is a key factor in the decision to use multi-agent systems. These challenges are addressed in Chapter **??**, in order to ensure the effectiveness of planning with reachability and co-observation in securing multi-agent systems.

We propose to form *sub-teams* on each route, and setup additional co-observations both within the sub-team and across different sub-teams. Notice that co-observations across sub-teams (named *cross-trajectory co-observation*) are always preferred when feasible, since they provide better security in potential situations where the entire sub-team is compromised. In this chapter, we consider only the high-level path planning, while the assignment of the duties in the team will be performed dynamically as introduced later in Chapter **??**), making it more difficult for attackers to successfully plan and execute attacks. Cross-trajectory co-observations allows sub-teams to preserve the optimal unsecured trajectories (as shown in Figure 1), as it does not require the entire *sub-team* to maintain a close distance with other teams when co-observation occurs.

a) Paper contributions: In this paper, we formulate a way to enforce an empty intersection between forbidden region and reachability regions, such that if an attacker takes control of the robots, they cannot perform an undetected attack by entering forbidden regions and meeting co-observation schedules at the same time. The constraints are formulated using an ellipsoidal bound of the reachability region.

We propose to incorporate redundancy in the form of *sub-teams* of multiple robots in place of individual robots along each route. This allows robots to deviate from their assigned sub-team, and join others to perform cross-trajectory co-observations, thereby securing multiple trajectories. We propose a formulation of the multi-flow problem on unsecured MRS trajectories to plan the cross-trajectory co-observations that can preserve the security against plan-deviation attacks.

II. Preliminaries

In this section, we review various mathematical concepts that will provide the foundations and context for our novel constraints.

A. Differentials

We define the differential of a map $f(x) : \mathbb{R}^m \to \mathbb{R}^n$ at a point x_0 as the unique matrix $\partial_x f \in \mathbb{R}^{n\times m}$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t} f\big(x(t)\big)\bigg|_{t=0} = \partial_x f\big(x(0)\big)\dot{x}(0) \tag{1}$$

where $t \mapsto x(t) \in \mathbb{R}^n$ is a smooth parametric curve such that $x(0) = x$ with any arbitrary tangent $\dot{x}(0)$. For later part of this paper, we will use $\dot{f}$ for $\frac{\mathrm{d}}{\mathrm{d}t} f$ and $\partial_x f$ for $\frac{\partial f}{\partial x}$. The differentials $\partial_x f$ is derived through (1) having $\dot{f}$ divided by $\dot{x}$.

With a slight abuse of notation, we use the same notation $\partial_x f$ for the differential of a matrix-valued function with scalar arguments $f : \mathbb{R}^R \to \mathbb{R}^{m\times n}$ Note that in this case (1) is still formally correct, although semantically different.

B. Alternating Directions Method of Multipliers (ADMM)

In this section, we briefly review the path planning algorithm based on ADMM constraints discussed in [7]. This algorithm can deal with a variety of types of non-convex and non-smooth optimization constraints that are commonly encountered in trajectory optimization tasks once they are formulated in a certain format. The goal of the section is to lay the groundwork for proposing our novel constraint in the following sections.

Let $x_{ij} \in \mathbb{R}^m$ denote the position of agent $i \in \{1, \dots, n\}$ at the discrete-time index $j \in [0, \dots, T]$, with m being the dimension of the workspace, n the number of agents, and T the planning time horizon. Let $\mathbf{x} = \mathrm{stack}(\{x_{ij}\})$ denote the aggregated vector of all the agents' trajectories over the entire time interval. The goal of the algorithm is to optimize an objective function $\Phi(\mathbf{x})$ on a set Ω defined by nonlinear constraints. To solve the problem using ADMM, we introduce a new set of variables $\mathbf{z} = D(\mathbf{x})$ to translate the constraints Ω on x to (typically lower-dimensional) sets of constraints on $\mathbf{z}$, such that the projection operator on ζ can be easily implemented (e.g., a constrained distance between two agents is translated into their difference constrained to an origin-centered sphere). The new set of constraints ζ on z is incorporated using an indicator function Θ_ζ. The main problem is written as:

$$\begin{aligned} \min \quad & \Phi(\mathbf{x}) + \Theta_\zeta(\mathbf{z}) \\ s.t. \quad & D(\mathbf{x}) - \mathbf{z} = 0, \end{aligned} \tag{2}$$

where $D(\mathbf{x}) = [D_1(\mathbf{x})^T, \dots, D_l(\mathbf{x})^T]^T$ is a vertical concatenation of different functions for different constraints.

And having the update steps shown as:

$$\mathbf{x}^{k+1} := \underset{\mathbf{x}}{\mathrm{argmin}}(\Phi(\mathbf{x}^k) + \frac{\rho}{2}\|D(\mathbf{x}) - \mathbf{z}^k + \mathbf{u}^k\|_2^2) \tag{3a}$$

$$\mathbf{z}^{k+1} := \Pi_\zeta(D(\mathbf{x}^{k+1}) + \mathbf{u}^k) \tag{3b}$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + D(\mathbf{x}^{k+1}) - \mathbf{z}^{k+1}, \tag{3c}$$

Since $D(\mathbf{x})$ is the vertical concatenation of $D_l(\mathbf{x})$, the projection of each set ζ_l is independent for each constraint and can be computed separately. The advantage of this formulation is that we can choose $D(\mathbf{x})$ such that the new constraint set ζ becomes simple to compute; however, the drawback is that we move the non-convexity to the primal cost function, i.e., in the update for $\mathbf{x}$ in (3a). Additionally, the function $D(\mathbf{x})$ can be used to select only the subset of the variables on which a constraint depends.

We consider the following types of traditional path planning constraints:

1) Velocity constraints: The movement of each agent is constrained by a maximum distance in any direction over a single discrete time step. We then define the function $D(\mathbf{x})$ to return the velocity vectors for the i-th agent at time step j:

$$D_{ij}(\mathbf{x}) = x_{ij} - x_{i(j-1)}, \quad j \in \{1, \dots, T\} \tag{4}$$

and the constraint set is:

$$\zeta_{ij} = \big\{ z \mid \|z\| \leq v_{max} \big\}. \tag{5}$$

The projection operator $\Pi_\zeta(z)$ for this constraint implies that the projection of the vector z is inside a sphere with a radius of v_{max} which can then be written as:

$$\Pi_\zeta(z) = \begin{cases} v_{max} \frac{z}{\|z\|} & \text{if } \|z\| > v_{max}, \\ z & otherwise. \end{cases} \tag{6}$$

This constraint could be adapted also to enforce more refined dynamical models.

2) Convex obstacles: : We use convex polygons to model regions defining the boundary of the workspace, as well as solid obstacles that cannot be entered by agents (note that the latter represent a non-convex constraint set from the point of view of the optimization problem). Obstacles with non-convex shapes can be modeled using unions of the convex sets.

We enforce the constraints at each discrete time step x_{ij} (enforcement of the constraints between these points can be achieved by making the obstacles sightly bigger than its actual size). For a convex area, waypoints stays unchanged it they are outside the zone, and if the waypoints are inside the zone, the constraint function $D(\mathbf{x})$ returns the least negative distance from the waypoints to the zone defined by hyperplanes which is represented as:

$$D_i(\mathbf{x}) = \begin{bmatrix} d_{11} \\ \vdots \\ d_{nm} \end{bmatrix} \tag{7}$$

where

$$d_{ij} = \max(\min(d_{ijk}, 0)), \tag{8}$$

$$d_{ijk} = p_{ij}^{\mathrm{T}} n_k - m_k, \tag{9}$$

and n_k is the normal vector and m_k is the scalar offset defining the k-th hyperplane. The corresponding constraint set is simply

$$\zeta = \{z \mid z = 0\}, \tag{10}$$

with a projection function:

$$\Pi_\zeta(z) = 0. \tag{11}$$

The motivating idea for (7) and (11) is to find waypoints inside the zone and project them to the closest boundary. As mentioned before, non-convex obstacles can be handled by the union of (possibly overlapping) convex obstacles.

3) Waypoints with flexible deadlines: In this type of constraint, we assume that an agent needs to go through a spherical ball around a given point p with a radius of d_{max} at some time instant j belonging to a given time window $[t_1, t_2]$. In this case, we define the function $D(\mathbf{q})$ to return the smallest distance from the point p to any point on the trajectory restricted to the relevant time window. This can be expressed precisely in the following form:

$$D_i(\mathbf{q}) = \min_{i \in \{1\dots,n\}, j \in \{t_1,\dots,t_2-1\}} \big(\mathrm{dist}(p, \overrightarrow{q_{ij} q_{i(j+1)}})\big), \tag{12}$$

with the constraint set:

$$\zeta = \{z \mid z < z_{max}\}, \tag{13}$$

where $\mathrm{dist}(p, \overrightarrow{q_{ij} q_{i(j+1)}})$ returns the distance between the fixed point p and the segment $\overrightarrow{q_{ij} q_{i(j+1)}}$. Note that this function returns the smallest distance between (p, q_{ij}) and $(p, q_{i(j+1)})$ if the projection of the point p does not lie on the line segment $\overrightarrow{q_{ij} q_{i(j+1)}}$; as a consequence, this constraint does not need to be satisfied exactly at one of the points on the discretized trajectory, but it can also be satisfied "en route" on the segment between them. The projection $\Pi_\zeta(z)$ for this constraint can be written as:

$$\Pi_\zeta(z) = \min(z, z_{max}). \tag{14}$$

Note that (13) and (14) are equivalent to (5) and (6), except for the fact that the quantity in (12) is always a positive scalar.

C. Householder rotations

To formulate the operator Π_ζ for our novel constraint, we will define a differentiable transformation of the constraint set to a canonical form. This transformation will include a rotation that we derive from a modified version of Householder transformations [8]. With respect to the standard definition, our modification ensures that the final operator is a proper rotation (i.e., not a reflection). We call our version of the operator a *Householder rotation*. In this section we derive Householder rotations and their differentials for the 3-D case; the 2-D case can be easily obtained by embedding it in the $z = 0$ plane.

Definition 1. *Let $\nu_\mathcal{F}$ and $\nu_\mathcal{E}$ be two unitary vectors ($\|\nu_\mathcal{F}\| = \|\nu_\mathcal{E}\| = 1$). Define the normalized vector u as*

$$u' = \nu_\mathcal{F} + \nu_\mathcal{E}, \quad u = \frac{u'}{\|u'\|}. \tag{15}$$

The Householder rotation *$H(\nu_\mathcal{F}, \nu_\mathcal{E})$ is defined as*

$$H(\nu_\mathcal{F}, \nu_\mathcal{E}) = 2uu^{\mathrm{T}} - I. \tag{16}$$

The main property of interest for our application is the fact that H is a rotation mapping $\hat{\nu_\mathcal{F}}$ to $\hat{\nu_\mathcal{E}}$, as shown by the following.

Proposition 1. *The matrix H has the following properties:*

1) It is a rotation, i.e.

a) $H^{\mathrm{T}}H = I$;
b) $\det(H) = 1$.

2) $\nu_{\mathcal{E}} = H\nu_{\mathcal{F}}$.

We compute the differential of H implicitly using the relation (1). We will use the notation $[v]_\times : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ to denote the matrix representation of the cross product with the vector v, i.e.,

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{17}$$

such that $[v]_\times w = v \times w$ for any $w \in \mathbb{R}^3$. One can verify by direct computation the following property:

$$wv^{\mathrm{T}} - vw^{\mathrm{T}} = [[v]_\times w]_\times. \tag{18}$$

Proposition 2. *Let $\nu_{\mathcal{F}}(t)$ represent a parametric curve. Then we have*

$$\dot{H} = H[-2M\dot{\nu}_{\mathcal{F}}]_\times \tag{19}$$

where the matrix $M \in \mathbb{R}^{3\times 3}$ is given by

$$M = [u]_\times \frac{\left(I - uu^{\mathrm{T}}\right)\left(I - \nu_{\mathcal{F}}\nu_{\mathcal{F}}^{\mathrm{T}}\right)}{\|u'\|\|\nu_{\mathcal{F}}\|}. \tag{20}$$

D. Rapidly-exploring Random Trees

We use RRT* as a component of the solution [**?**], which is an optimized variant of the rapidly-exploring Random Trees (RRT). As a optimal path planning algorithm, RRT* returns the shortest paths between an initial location and multiple potential goals. We assume that the generated paths can be travelled in both directions (this is used later in our analysis). The basic RRT* algorithm is shown in Algorithm **??**. For a given tree $G = (V, E)$, key functions called by the algorithm are

`Sample`(i) This function returns independent identically distributed samples from the free configuration space.

`Nearest`(x) This function returns the vertex $v \in V$ that is closest in Euclidean distance to the input point x.

`Steer`(x, y) This function returns a point z that minimize $\|z - y\|$ subject to $\|z - x\| \leq \eta$ for a prespecified $\eta > 0$.

`ObstacleFree`(x, y) This function returns whether the line segment between x and y is free of collision.

`Cost`(v) This function assigns a non-negative cost (total travel distance in our application) to the unique path from the initial position to v .

`Parent`(v) This is a function that maps the vertex v to $v' \in V$ such that $(v', v) \in E$

III. Secure Multi-robot path planning using co-observation and reachability constraints

Besides traditional path planning constraints, additional security constraints need to be incorporated into the path planning problem to prevent cases where an attacker changes a robot's path between two observations to reach a forbidden region without being detected. The work of [5] provide exact solution for grid-world configuration. Using the onboard sensing capabilities, robots can perform inter-robot observations as an additional security measure. Each robot' self-report includes not only its own status, but also additional observations of other robots' status. For example, robot i's self-report may include its observation of robot j, "i report its arrival at v and i observes j at location w". Similarly, robot j also reports its observations of robot i to the CE. This solution provides paths with an observation schedule such that any attempts by a compromised robot to violate the safety constraints would necessarily break the observation plan and be detected.

When transforming the solution to continuous configuration spaces, additional analysis is needed on the reachability of robots in order for the security of co-observation schedule to hold. More broadly, the problem of solving a path optimization problem with constraints based on the sets of locations that the agents could *potentially* reach, which we call *reachability regions*, has not received attention in the literature. This constraint is formulated using an ellipsoidal bound of the reachability region, and the idea of using an ellipsoidal bound to limit the search space is inspired by the *heuristic sampling domain* introduced by [6] in the context of the RRT* path planning algorithm.

We formulate a way to enforce an empty intersection between forbidden regions and ellipsoidal reachability region while optimizing the trajectory, such that if an attacker takes control of the robots, they cannot perform an undetected attack without breaking the co-observation schedule. To the best of our knowledge, our path planning algorithm is the only one flexible enough to handle the safety related spatio-temporal constraints introduced by co-observations and reachability.

A. Co-observation schedule

The co-observation constraints is used to guarantee that, at time required by the schedule, two robots should get close enough with each other to observe each other's behavior. The co-observation constraint is modeled as a relative distance constraint between two robots at some time instant j (i.e., to require two agents see each other within a certain radius to inspect each other, or to exchange data). The location of the co-observation location should make sure that the ellipsoidal reachability region between each co-observation, considered in later sections III-B, has an empty intersection with all forbidden regions. We write the function for the constraint as:

$$D(\mathbf{q}) = \overrightarrow{q_{aj}q_{bj}}, \tag{21}$$

where a, b are the indices of the pair of agents required for a mutual inspection. The corresponding constraint set is simply

$$\zeta = \{z \mid z \leq d_{max}\}, \tag{22}$$

with a projection function:

$$\Pi_\zeta(z) = \begin{cases} d_{max}\frac{z}{\|z\|} & \text{if } \|z\| > d_{max}, \\ z & otherwise. \end{cases} \tag{23}$$

The locations q_{aj} and q_{bj} where the co-observation performed are computed as part of the optimization.

B. Reachability constraints

In this section, we first provide the definition of an *ellipsoidal reachability region*. We then provide a differentiable map

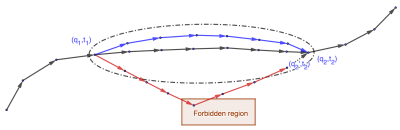


Fig. 2: The ellipse showcases the reachability region. The black line is the planned trajectory of a robot, q_1 and q_2 are two locations this robot are expected at given time t_1 and t_2, blue and red line are possible trajectories if the robot is compromised in after reaching q_1. Robot that goes outside the reachability region to forbidden region can only reach q_2' instead of q_2 at time t_2, even if it drives directly toward q_2 at full speed v_{max} after the intrusion into forbidden region.

to transform such region in a canonical axis-aligned form where the operator Π_ζ and its differential can be obtained; these operators are then extended to the general case via the aforementioned transform. The overall goal is to define the functions $D(q)$, its differential, and the operator Π_ζ for *ellipsoidal reachability regions* with respect to the forbidden regions that can then be used in the ADMM formulation in Chapter **??**.

C. Definition of reachability constraint

The reachability region is defined as the set of locations $q(t)$ that a robot can reach between two given fixed positions:

Definition 2. *The* reachability region *for two waypoints $q(t_1) = q_1$, $q(t_2) = q_2$ is defined as the sets of points q' in the workspace such that there exist a trajectory $q(t)$ where $q(t') = q'$, $t_1 \leq t' \leq t_2$ and $q(t)$ satisfies the velocity constraint $d(q(t), q(t+1)) \leq v_{max}$.*

This region can be analytically bounded via an ellipsoid:

Definition 3. *The* reachability ellipsoid *is defined as the region $\mathcal{E}(q_1, q_2, t_1, t_2) = \{\tilde{q} \in \mathbb{R}^n : d(q_1, \tilde{q}) + d(\tilde{q} + q_2) < 2a\}$, where $a = \frac{v_{maq}}{2}(t_2 - t_1)$.*

The region $\mathcal{E}(q_1, q_2)$ is an ellipsoid with foci at q_1, q_2, center $o_\mathcal{E} = \frac{1}{2}(q_1 + q_2)$, and the major radius equal to a. Let $c_\mathcal{E} = \frac{1}{2}\|q_1 - q_2\| = \|o_\mathcal{E} - q_1\|$ be the distance from the center to a foci.

The reachability ellipsoid is an over-approximation of the exact reachability region; the difference between the two is due to the discretization of the trajectory, and the fact that $\mathcal{E}$ does not consider the presence of obstacles.

D. Transformation to canonical coordinates

To simplify the problem, a canonical rigid body transformation is used to transform the ellipse $\mathcal{E}$ from a global frame $\mathcal{F}$ to a canonical frame $\mathcal{F}_\mathcal{E}$. The latter is defined such that the center of the ellipsoid is located at the origin and the foci are aligned with the first axis of $\mathcal{F}_\mathcal{E}$. Since the transformation depends on the two foci, i.e., two waypoints of an agent, the challenge here is to construct the transformation in a differentiable way. For the convenience of derivation, we define the coordinate transformation from $\mathcal{F}$ to $\mathcal{F}_\mathcal{E}$ using a rotation $R_\mathcal{E}^\mathcal{F}$ and a translation $o_\mathcal{E}^\mathcal{F}$, which, to simplify the notation, from now on we simply refer to as R and o, respectively. The transformation of a point from the frame $\mathcal{F}_\mathcal{E}$ to the frame $\mathcal{F}$ and its inverse are given by the formulation:

$$q^\mathcal{F} = Rq^\mathcal{E} + o, \quad q^\mathcal{E} = R^\mathrm{T}(q^\mathcal{F} - o). \tag{24}$$

The reverse transformation is $q^\mathcal{E} = R^\mathrm{T}(q^\mathcal{F} - o)$.

We define $\nu_\mathcal{F}$ and $\nu_\mathcal{E}$ to represent the x-axis unitary vector of $\mathcal{F}_\mathcal{E}$ in the frames $\mathcal{F}$ and $\mathcal{F}_\mathcal{E}$, respectively. Formally:

$$\nu'_\mathcal{F} = q_2 - q_1, \quad \nu_\mathcal{F} = \frac{\nu'_\mathcal{F}}{\|\nu_\mathcal{F}\|}, \quad \nu_\mathcal{E} = [1, 0, 0]^T; \tag{25}$$

see Fig.2 for an illustration. Note that $\nu_\mathcal{E}$ is constant while, for the sake of clarity, $\nu_\mathcal{F}$ depends on q_1, q_2. We then define the rotation R using a Householder rotation, while from (24) we see that o represents the center of $\mathcal{E}(q_1, q_2)$ expressed in $\mathcal{F}$, i.e.:

$$R = H(\nu_\mathcal{F}(q_1, q_2), \nu_\mathcal{E}(q_1, q_2)), \quad o = \frac{1}{2}(q_1 + q_2). \tag{26}$$

To simplify the notation, in the following we will consider H to be a function of q_1, q_2 directly, i.e. $H(q_1, q_2)$.

E. Reachability constraints via ellipsoids

In later sections, we define constraint formulation of the ellipsoid regions against different types of forbidden regions: a point, a plane, a segment, and a convex polygon. For each one, our goal is to define the function $D(q)$, the set ζ and the projection Π_ζ that can be incorporated in the ADMM optimization (**??**); we also include derivations for the differential $\partial_q D(q)$, which can be used to significantly speed up the optimization.

F. Point-ellipsoid constraint

As shown in Fig.3, we consider a forbidden region in the shape of a single point q_{avoid}. The goal is to design the trajectory $q(t)$ such that $q_{avoid} \notin \mathcal{E}(q_1, q_2)$. We first define a projection function $\pi_{p\mathcal{E}}(q_{avoid}; q_1, q_2, a) = q_p$, which returns a projected point q_p of q_{avoid} outside the ellipse, i.e., as the solution to

$$\begin{aligned} \underset{q_p}{\text{argmin}} \quad & \|q_{avoid} - q_p\|^2 \\ s.t. \quad & q_p \in \mathcal{E}^c. \end{aligned} \tag{27}$$

where $\mathcal{E}^c$ is the set complement of the region $\mathcal{E}$.

Then, the constraint $q_{avoid} \notin \mathcal{E}(q(t_1), q(t_2))$ can be written as:

$$D(q) = \pi_{p\mathcal{E}}(q_{avoid}; q(t_1), q(t_2), r) - q_{avoid} = 0 \tag{28}$$

For cases where $q_{avoid} \notin \mathcal{E}(q(t_2), q(t_1), r)$, $\pi_{p\mathcal{E}}(q_{avoid}) = q_{avoid}$. And for cases where $q_{avoid} \in \mathcal{E}(q_1, q_2, r)$, $D(q)$ needs to be projected to the boundary of the ellipse, which is discussed below.

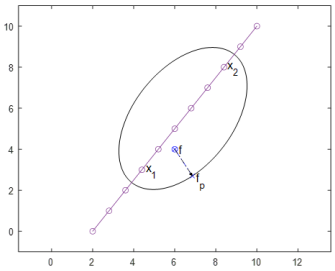


Fig. 3: Point-ellipsoid constraint, a point f inside ellipsoid is projected to the areas outside the ellipsoid f_p.

1) Projection to the standard ellipse: The ellipse $\mathcal{E}$ expressed in $\mathcal{F}_\mathcal{E}$ is given by $\mathcal{E}^\mathcal{E} = \{q^\mathcal{E} \in \mathbb{R}^m : d(q_1^\mathcal{E}, q^\mathcal{E}) + d(q^\mathcal{E}, q_2^\mathcal{E}) < 2a\}$, where the coordinates of the two foci $q_1^\mathcal{E}, q_2^\mathcal{E}$ in $\mathcal{F}_\mathcal{E}$ are

$$q_1^\mathcal{E} = \begin{bmatrix} c & 0 & 0 \end{bmatrix}^\mathrm{T}, \qquad q_2^\mathcal{E} = \begin{bmatrix} -c & 0 & 0 \end{bmatrix}^\mathrm{T}, \tag{29}$$

with $c = \frac{\|q_2 - q_1\|}{2}$.

The ellipsoid $\mathcal{E}$ in the canonical frame can be described as the zero level set of the quadratic function

$$E^\mathcal{E}(q^\mathcal{E}) = {q^\mathcal{E}}^\mathrm{T} Q q^\mathcal{E} - 1 \tag{30}$$

where

$$Q = \mathrm{diag}(a^{-2}, b^{-2}, b^{-2}), \tag{31}$$

and $b = \sqrt{a^2 - c^2}$. The ellipse parameters a,b represent the lengths of the major axes.

The point to project, q_{avoid}, can be likewise expressed in $\mathcal{F}_\mathcal{E}$ as ${q^\mathcal{E}}_{avoid} = H({q^\mathcal{F}}_{avoid} - o)$.

We now turn our attention to the problem of projecting ${q^\mathcal{E}}_{avoid}$ on the zero level set of $E^\mathcal{E}$ (i.e., the reachability ellipsoid in the canonical frame). The derivations below are loosely inspired by [**?**].

Let $q_p^\mathcal{E}$ be the point on the surface of the ellipsoid, i.e., $E^\mathcal{E}(q_p^\mathcal{E}) = 0$, corresponding to the projection of the point $q_{avoid}^\mathcal{E}$. Using Lagrange multipliers applied to the constrained optimization problem (27) (after transforming it in the canonical frame), one can show that the vector from a point to its projection, $q_{avoid}^\mathcal{E} - q_p^\mathcal{E}$, must be collinear with the gradient of $\mathcal{E}^\mathcal{E}$, i.e.

$$q_p^\mathcal{E} - q_{avoid}^\mathcal{E} = s\partial_q E^\mathcal{E}(q_p^\mathcal{E})^\mathrm{T} = sQq_p^\mathcal{E} \tag{32}$$

for some scale $s \in \mathbb{R}$; thus $q_p^\mathcal{E}$ can be written as:

$$q_p^\mathcal{E} = (I + sQ)^{-1} q_{avoid}^\mathcal{E} = S q_{avoid}^\mathcal{E} \tag{33}$$

where $S = (I + sQ)^{-1}$. Using the fact that since $q_p^\mathcal{E}$ is a point on the ellipse, s can be solved as the root of the equation obtained by substituting (33) in $E^\mathcal{E}(q^\mathcal{E})$:

$$0 = F(s) = {q_p^\mathcal{E}}^\mathrm{T} Q q_p^\mathcal{E} - 1 = {q_{avoid}^\mathcal{E}}^\mathrm{T} Q'(s) q_{avoid}^\mathcal{E} - 1, \tag{34}$$

where

$$Q'(s) = S^\mathrm{T} Q S = \mathrm{diag}\left(\frac{a^2}{(s+a^2)^2}, \frac{b^2}{(s+b^2)^2}, \frac{b^2}{(s+b^2)^2} \right) \tag{35}$$

Detailed methods for computing s can be found in [**?**].

Then the point-to-ellipse projection function can be represented as:

$$\begin{aligned} \pi_{p\mathcal{E}}(q) &= R^{-1}(q(t_1), q(t_2)) q_p^\mathcal{E} + o \\ &= R^{-1}(q(t_1), q(t_2)) S q_{avoid}^\mathcal{E} + o \\ &= R^{-1} S R (q_{avoid} - o) + o \end{aligned} \tag{36}$$

In our derivations, we consider only the 3-D case ($m = 3$); for the 2-D case, let $P = \begin{bmatrix} I & 0 \end{bmatrix} \in \mathbb{R}^{2\times 3}$: then $\pi_{p\mathcal{E}}^\mathrm{2D} = P \pi_{p\mathcal{E}}^\mathrm{3D}(P^\mathrm{T} q_{avoid}; P^\mathrm{T} q_1, P^\mathrm{T} q_2, a)$.

2) ADMM constraints: The corresponding constraint is written as

$$D_p(q) = \begin{cases} \pi_{p\mathcal{E}}(q) - q_{avoid} & q_{avoid} \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

and feasible set and projection function as:

$$\zeta = \{q \in \mathbb{R}^{nm} : \|D_p(q)\| = 0\}, \quad \Pi(D_p(q)) = 0 \tag{38}$$

Proposition 3. *The differential of the projection operator $\pi_{p\mathcal{E}}(q_{avoid}; q_1, q_2, a)$ with respect to the foci q_1, q_2 is given by the following (where we use q as a shorthand notation for $q_{avoid}^\mathcal{E}$)*

$$\begin{aligned} \partial_{\left[\begin{smallmatrix} q_1 \\ q_2 \end{smallmatrix}\right]} \pi_{p\mathcal{E}} &= -2H[SH(q-o)]_\times U \\ &+ \big((q^\mathrm{T} \partial_s Q' q)^{-1} H^{-1} Q' q q^\mathrm{T} (4Q' H[q-o]_\times U \\ &+ 2Q' H \partial_q o - \partial_b Q' q q \partial_q b) - s H^{-1} S^2 \partial_b Q q \partial_q b \big) \\ &- 2H^{-1} S H [q-o]_\times U + (H^{-1} S H - I) \partial_q o \end{aligned} \tag{39}$$

The differential of D_p is the same as the one for $\pi_{p\mathcal{E}}$.

G. Plane-ellipsoid constraint

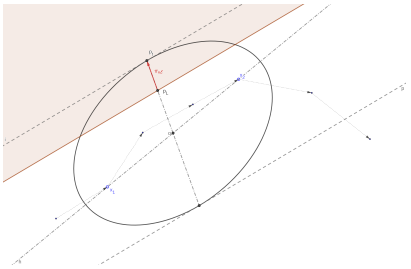


Fig. 4: Plane-ellipse constraint. Projected a ellipsoid to one side of a plane. The projection is simplified to the point-ellipse constraint as projecting a point inside ellipse p_L to p_t on outside the ellipse.

For a intermediate step to get the relationship between a polygon shaped forbidden region, we first consider a forbidden region in the shape of a hyperplane $\mathcal{L}(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^n : \mathbf{n}^\mathrm{T} \tilde{q} = d\}$ (as shown in Figure 4). The reachability constraint now can then be defined as $\mathcal{L} \cap \mathcal{E}(q_1, q_2, a) = \emptyset$. Using the transformation introduced in (III-D), the ellipse can be transferred into a standard one with the new hyperplane in the form of $\mathcal{L}^\mathcal{E}(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_\mathcal{E}^\mathrm{T} \tilde{q} = d_\mathcal{E}\}$, where,

$$n_\mathcal{E} = H(q_1, q_2) n \tag{40}$$

$$d_\mathcal{E} = -\mathbf{n}^\mathrm{T} o + d \tag{41}$$

1) Projection via the tangent interpolation point: For each $\mathcal{L}_\mathcal{E}$, there exist two planes ${\mathcal{L}^\mathcal{E}}_1 = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_\mathcal{E}^\mathrm{T}\tilde{q} = d_{\mathcal{E}t}\}$ and ${\mathcal{L}^\mathcal{E}}_2 = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_\mathcal{E}^\mathrm{T}\tilde{q} = -d_{\mathcal{E}t}\}$ that are parallel to $\mathcal{L}$ and tangent to the ellipse, where

$$d_{\mathcal{E}t} = \sqrt{n_\mathcal{E}^\mathrm{T} Q^{-1} n_\mathcal{E}}. \tag{42}$$

The corresponding tangent points are given as

$$p_1^\mathcal{E} = \frac{d_{\mathcal{E}t} Q^{-1} n_\mathcal{E}}{n_\mathcal{E}^\mathrm{T} Q^{-1} n_\mathcal{E}} = \frac{Q^{-1} n_\mathcal{E}}{d_{\mathcal{E}t}}, \tag{43}$$

$$p_2^\mathcal{E} = -p_1^\mathcal{E}. \tag{44}$$

We use these two tangent points to define a novel measure of displacement between a plane and an ellipse.

Definition 4. *The* tangent interpolation point *is defined as*

$$p_\mathcal{L}^\mathcal{E} = \frac{d_\mathcal{E} Q^{-1} n_\mathcal{E}}{n_\mathcal{E}^\mathrm{T} Q^{-1} n_\mathcal{E}} \tag{45}$$

Note that $p_\mathcal{L}^\mathcal{E} \in \mathcal{L}$ and when $d_\mathcal{E} = d_{\mathcal{E}t}$ or $d_\mathcal{E} = -d_{\mathcal{E}t}$, $p_\mathcal{L}^\mathcal{E} = p_1^\mathcal{E}$ or $p_\mathcal{L}^\mathcal{E} = p_2^\mathcal{E}$, respectively. It is clear that when $d_\mathcal{E} \in [-d_{\mathcal{E}t}, d_{\mathcal{E}t}]$, the plane $\mathcal{L}$ and the ellipsoid $\mathcal{E}$ have at least one intersection, thus violating our desired reachability constraint. Using this fact, we define the following projection operator:

$$\pi_{\mathbf{n}\mathcal{E}}^\mathcal{E}(q) = \begin{cases} p_{t1}^\mathcal{E} - p_\mathcal{L}^\mathcal{E} & d_\mathcal{E} \in [0, d_{\mathcal{E}t}], \\ p_{t2}^\mathcal{E} - p_\mathcal{L}^\mathcal{E} & d_\mathcal{E} \in [-d_{\mathcal{E}t}, 0), \\ 0 & \text{otherwise.} \end{cases} \tag{46}$$

2) ADMM constraints: Transforming the projection $\pi^\mathcal{E}$ back to $\mathcal{F}$ we have:

$$D_\mathbf{n}(q) = H^{-1}(q(t_1), q(t_2)) \pi_{\mathbf{n}\mathcal{E}}^\mathcal{E}(q) + o, \tag{47}$$

with the corresponding constraint written as

$$\zeta_\mathbf{n} = \{q \in \mathbb{R}^{nm} : \|D_\mathbf{n}(q)\| = 0\} \tag{48}$$

$$\Pi_\mathbf{n}(D_\mathbf{n}(q)) = 0 \tag{49}$$

3) Gradients for the projection function:

Proposition 4. *The differential of the projection function* ${\Pi^\mathcal{E}}_{\mathbf{n}\mathcal{E}}(q)$ *with respect to the foci* q_1 *and* q_2 *is given by:*

$$\partial_q \pi_{\mathbf{n}\mathcal{E}}^\mathcal{E}(q) = \begin{cases} \partial_q p_{t1}^\mathcal{E} - \partial_q p_\mathcal{L}^\mathcal{E} & d_\mathcal{E} \in [0, d_{\mathcal{E}t}], \\ \partial_q p_{t2}^\mathcal{E} - \partial_q p_\mathcal{L}^\mathcal{E} & d \in [-d_{\mathcal{E}t}, 0), \\ 0 & otherwise. \end{cases} \tag{50}$$

where

$$\partial_q p_\mathcal{L} = (-\frac{d_{\mathcal{E}t} n^\mathrm{T} \partial_q o - 2 d_\mathcal{E} \partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^3}) Q^{-1} n_\mathcal{E} + \frac{d_\mathcal{E} \partial_b Q^{-1} n_\mathcal{E} \partial_q b - 2 d_\mathcal{E} Q^{-1} H[n]_\times U}{d_{\mathcal{E}t}^2}, \tag{51}$$

$$\partial_q p_1 = -\frac{Q^{-1} n_\mathcal{E} \partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^2} + \frac{\partial_b Q^{-1} n_\mathcal{E} \partial_q b - 2 Q^{-1} H[n]_\times U}{d_{\mathcal{E}t}}. \tag{52}$$

Based on the previous proposition, the differential of (47) can be written as:

$$\partial_q D_{\mathbf{n}\mathcal{E}} = -2H[{\Pi^\mathcal{E}}_{\mathbf{n}\mathcal{E}}]_\times M + H^{-1} \partial_q {\Pi^\mathcal{E}}_{\mathbf{n}\mathcal{E}} \tag{53}$$

H. Line-segment-ellipse constraint

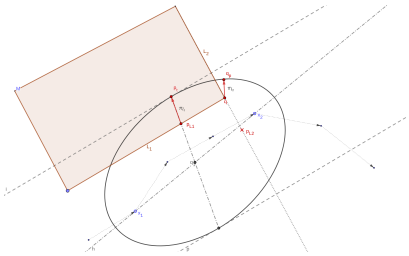


Fig. 5: Polygen-ellipse constraint. This constraint is simplified to either a plane-ellipse constraint or a point-ellipse constraint.

To avoid (or stay in) a region defined by several hyperplanes, the relative position between the ellipse and segment of the hyperplane needs to be studied instead of the whole plane. Assume the segment has endpoints p_1 and p_2, the segment can be defined as:

$$\begin{bmatrix} (p_1 - p_2)^\mathrm{T} \\ (p_2 - p_1)^\mathrm{T} \end{bmatrix} p \leq \begin{bmatrix} p_2^\mathrm{T} \\ -p_1^\mathrm{T} \end{bmatrix} (p_1 - p_2) \tag{54}$$

And using the $p_\mathcal{L}$ defined in (45), we define

$$D_{seg}(q) = \begin{cases} D_1(p_1) & (p_1 - p_2)^\mathrm{T}(p_\mathcal{L} - p_2) < 0 \\ D_1(p_2) & (p_2 - p_1)^\mathrm{T}(p_\mathcal{L} - p_1) < 0 \\ D_2(p_\mathcal{L}) & otherwise \end{cases} \tag{55}$$

where $D_{p\mathcal{E}}$ and $D_{\mathbf{n}\mathcal{E}}$ are corresponding constraint function introduced in section III-F and III-G with the constraint set and projection written as:

$$\zeta_{seg} = \{q \in \mathbb{R}^{nm} : \|D_{seg}(q)\| = 0\} \tag{56}$$

$$\Pi_{seg}(D_{seg}(q)) = 0 \tag{57}$$

I. Convex-polygon-ellipse constraint

To keep an ellipse away from a convex polygon, first, we need to keep all segments of the hyperplanes outside the ellipse. And a convex obstacle constraint for the polygon (as introduced in section **??**) is also added to prevent cases where the ellipse is a subset of the region. Similar to (45), we define

$$D_{cp}(q) = \begin{bmatrix} D_{seg1} \\ D_{seg2} \\ \vdots \end{bmatrix} \tag{58}$$

with the constraint set and projection written as:

$$\zeta_{cp} = \{q \in \mathbb{R}^{nm} : \|D_{cp}(q)\| = 0\} \tag{59}$$

$$\Pi_{cp}(D_{cp}(q)) = 0 \tag{60}$$

IV. Cross-trajectory co-observation planning

Assume we have a total of N robots in the system. We define a partition $\mathcal{I} = \cup_p \mathcal{I}_p$ of the robots such that robots in *sub-team* $\mathcal{I}_p$ have the same nominal trajectory. We use the planner introduced in Chapter **??** to generate multi-robot trajectories with $N_p < N$ routes $\{q_p\}_{p=1}^{N_p}$ without reachability and co-observation constraints, where the state $q_p(t), p \in \{0, \ldots, N_p\}$

represents the reference position of the i-th sub-team at time $t \in \{0, \ldots, t_e\}$. Within each sub-team, at least one robot is required to follow the reference trajectory to fulfill the required tasks. Meanwhile, the additional robots can switch from one reference trajectory to another in order to perform co-observation with different sub-teams. We refer to the new type of co-observations as *cross-trajectory co-observations*. The goal is to find the cross trajectory co-observation plan to secure an unsecured MRS trajectory as with a minimal number of additional robots.

To solve this problem, we model the planned trajectory $\{q_p\}_{p=1}^{N_p}$ as a directed *checkpoint graph* $G_q = (V_q, E_q)$. Vertices V_q are a key locations in $\{q_p\}$ that are used to perform co-observations. Part of these vertices are *checkpoints* which are selected to guarantee that, if robots deviate and reach forbidden regions, they will miss the scheduled co-observations. Other vertices are selected to connect checkpoints on different trajectories. The set $E_q = E_t \cup E_c$ consists of directed edges of two types, trajectory edges E_t and cross-trajectory edges E_c, representing paths which robots can take. The trajectory edges E_t represent the reference trajectories, where each edge connects two waypoints on the trajectory of the same sub-team. The cross-trajectory edges E_c connect vertices on different reference trajectories, with at least one of the vertices being a checkpoint. These edges represent the potential paths that robots can take to travel and perform co-observations between trajectories of different sub-teams. Similar to the work done by [**?**], we show that additional robots in the sub-team can be formulated as flows in the checkpoint graph, transforming the co-observation planning problem into a network multi-flow problem that can be solved using general linear programming techniques as specialized solver.

In this chapter, we present a solution to the cross-trajectory co-observation planning problem. This approach consists of two components: constructing the checkpoint graph based on unsecured multi-robot trajectories, which includes the location of security checkpoints and cross-trajectory paths, and the formulation of the network flow problem to solve for the co-observation plan.

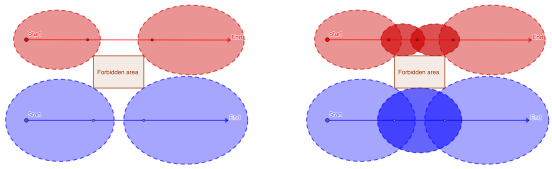
V. Checkpoint Graph Construction

In this section, we describe in detail how we define and search for security checkpoints and use the checkpoints to find cross-trajectory edges and construct a directed security graph $G_q = (V_q, E_q)$.

A. Checkpoints

Let $q \in \mathbb{R}^{nmT}$ be the MRS trajectory found in Chapter **??** for m sub-teams with time horizon $T = t_e$, with $q_p(t)$ be the reference trajectory waypoint for sub-team p at time t. A checkpoint $v_i = (q_i, t_i)$ is a pair composed of a location $q_i = q_p(t_i)$ and a time t_i representing a co-observation between a sub-team $\mathcal{I}_p$'s member and robots either from the same sub-team or different ones. For convenience, let $\mathcal{I}_{v_i}$ represent the corresponding sub-team v_i belongs to.

Similar to the requirements of co-observations introduced in Chapter **??**, checkpoints $V_p = \{v_{p_1}, \ldots, v_{p_e}\} \subset V_q$ for a reference trajectory $\{q_p(t)\}$ of sub-team p need to guarantee that the reachability region between consecutive checkpoints $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}})$ does not intersect with any of the forbidden areas. For convenience, we denote the union of all forbidden areas as F; then the requirement can be written as $\mathcal{E}(q_{p_i}, q_{p_{i+1}}, t_{p_i}, t_{p_{i+1}}) \cap F = \emptyset$, for every i.



(a) (b)

Fig. 6: Checkpoint generation example

We provide a heuristic approach to locate the checkpoints on given trajectories; an optimal solution would likely be NP-hard, while the approach below works well enough for our purpose. We begin by adding the start and end locations $q_p(t_0)$, $q_p(t_e)$ to V_q, where $t_0 = 0$, and t_e is the time horizon. Then, we search for the waypoint $q_p(t_1)$ with the largest $t_1 \in \{t_0, \ldots, t_2\}$ such that the reachability ellipsoid between $q_p(t_0)$ and $q_p(t_1)$ has no overlap with any of the forbidden areas, i.e. $\mathcal{E}(q_p(t_0), q_p(t_1), t_0, t_1) \cap F = \emptyset$. Simultaneously, we search backward to find $q_p(t_3)$ with the smallest $t_3 \in \{t_1, \ldots, t_2\}$ that meets the reachability requirement $\mathcal{E}(q_p(t_2), q_p(t_3), t_2, t_3) \cap F = \emptyset$. $q_p(t_1)$ and $q_p(t_3)$ are then added to V_q. Afterwards, we set $t_0 = t_1$, $t_2 = t_3$ and repeat the same process. The search is stopped when $t_0 > t_2$, or when $\mathcal{E}(q_p(t_0), q_p(t_2), t_0, t_2) \cap F = \emptyset$. A toy example is shown in Figure 6.

B. Cross-trajectory edges

After locating checkpoints $\cup_p V_p$ for all sub-teams, we can search for cross-trajectory edges to connected all checkpoints to trajectories. Cross-trajectory edges represents feasible paths between two references trajectories that allow robots to reach a different reference trajectories and perform co-observations with the corresponding sub-teams. Thus, at least one end of the edges must be a security checkpoint $v_p \in \cup_p V_p$. Additionally, assuming that only one robot is sent at a time between trajectories, the cross-trajectory edges must satisfy the reachability constraints from Chapter **??** to ensure that no deviation to forbidden areas can be performed while switching trajectories.

To find cross-trajectory edges, we base our approach on RRT*. First, we search for feasible paths between each security checkpoint in each reference trajectory and all other trajectories, as shown in Figure **??**. Since paths returned by RRT* are optimal and bidirectional, we can analyze the travel time and reachability for each candidate cross-trajectories, and add these edges to the checkpoint graph if they are feasible.

More vigorously, After a fixed iterations of RRT*, we find all possible and close to optimal paths between one security checkpoint $v_p = (q_p, t_p) \in V_p$ for sub-team $\mathcal{I}_p$, and

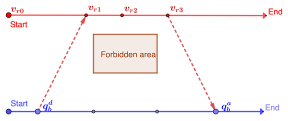


Fig. 7: For checkpoints of the red trajectory, latest departure node q_b^d found for v_{r1} and earliest arrival node q_b^a found for v_{r3}.

$V_r = \{(q_{r_i}, t_{r_i})\}$ where $q_{r_i} = q_r(t_{r_i})$ are the entire waypoints on a reference trajectories for a different sub-team $\mathcal{I}_r$ for a fixed number of RRT* iterations. We can calculate the minimal travel time $t = \texttt{Cost}(q)/v_{max}$ for a robot to traverse each path. We call all nodes $v_{r_i} \in V_r$ *arrival node*s if there exist a path between v_{r_i} and v_p with $t_p + t < t_{r_i}$, and the reachability ellipsoid $\mathcal{E}(q_r(t_{r_i}), q_p, t_{r_i}, t_p) \cap F = \emptyset$. Similarly, *departure node*s are defined for v_{r_i} with $t_p > t_{r_i} + t$ and $\mathcal{E}(q_p, q_r(t_{r_i}), t_p, t_{r_i}) \cap F = \emptyset$. Given a point $v_p \in V_p$, we define the *latest departure node* $q_r(t_r^d)$, and the *earliest arrival node* $q_r(t_r^a)$ as follows:

Latest departure node $q_r(t_r^d)$: is the waypoint with the latest time t_r^d such that a robot from sub-team $\mathcal{I}_r$ can meet with robots in sub-team $\mathcal{I}_p$ at v_p.

Earliest arrival node $q_r(t_r^a)$: is waypoint with the earliest time t_r^a such that a robot from sub-team $\mathcal{I}_p$ deviating from v_p can meet with sub-team $\mathcal{I}_r$.

For $v_p \in V_p$, latest departure node $v_d = (q_r(t_r^d), t_r^d)$ and earliest arrival node $v_a = (q_r(t_r^a), t_r^a)$, if feasible, are added to V_q. Cross-trajectory edges $(v_p \to v_a)$ and $(v_d \to v_p)$ are added to E_q. Examples are shown in Figure 7. Full algorithm is shown in Algorithm 1.

C. In-trajectory edges

All checkpoints V_q found in Section V-A and V-B, can be divided by the corresponding sub-team and sorted in ascending order of time into N_p subsets $V_p = \{v_0^p, v_1^p, \ldots, v_T^p\}$ where $p \in \{1, \ldots, N_p\}$. Consecutive edges in the same subset are added to E_q as in-trajectory edges $(v_i^p \to v_{i+1}^p)$. Example are shown in Figure 8.

VI. Co-observation planning problem

In this section, we describe in detail how to formulate the cross-trajectory planning problem as a network flow problem, and solve it using mixed-integer linear programs (MILP). We assume that there are always *reference robot*s in each sub-team following the reference trajectory. We want to plan the routes for additional *cross-trajectory robots* to perform co-observation with the reference robots.

To formulate the planning problem as a network multi flow problem, we augment the checkpoint graph G_q to a flow graph $G = (V, E)$. The vertices of the new graph are defined as $V = V_q \cup \{v^+, v^-\}$, a *virtual source* v^+ and a *virtual sink* node v^- are added to V. The edges of the new graph are defined

Algorithm 1 Cross-trajectory edges generation

procedure CrossTrajectoryEdges(V)
  $V_q \leftarrow V$; $E_q \leftarrow 0$;
  for all $v \in V$ **do**
    for all $\mathbf{q}_r$ that $\mathcal{I}_r \neq \mathcal{I}_v$ **do**
      $V_q, E_q \leftarrow$ AddEdges($q_v, \mathbf{q}_r, V_q, E_q$);
    end for
  end for
  return V_q, E_q
end procedure
procedure AddEdges($q_v, \mathbf{q}_r, V_q, E_q$)
  $t_r^d \leftarrow -\infty$; $t_r^a \leftarrow \infty$;
  $\mathcal{V}_{goal} = RRT^*(q_v, \mathbf{q}_r)$;
  for all $q \in \mathcal{V}_{goal}$ that $\texttt{Parent}(q) \neq \emptyset$ **do**
    if $\texttt{Cost}(q)/v_{max} < t_q - t_v$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ **then**
      if $t_q < t_r^a$ **then**
        $t_r^a \leftarrow t_q$;
      end if
    else if $\texttt{Cost}(q)/v_{max} < t_v - t_q$ and $\mathcal{E}(q, v_p, t_q, t_{v_p}) \cup F = \emptyset$ **then**
      if $t_q > t_r^d$ **then**
        $t_r^d \leftarrow t_q$;
      end if
    end if
  end for
  if $t_r^d \neq -\infty$ and $t_r^a \neq \infty$ **then**
    $v_a \leftarrow (q(t_r^a), t_r^a)$; $v_d \leftarrow (q(t_r^d), t_r^d)$;
    $V_q \leftarrow V_q \cup v_a$; $V_q \leftarrow V_q \cup v_d$;
    $E_q \leftarrow E_q \cup (v_p \to v_a)$; $E_q \leftarrow E_q \cup (v_d \to v_p)$
  end if
  return V_q, E_q
end procedure

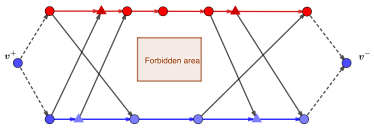


Fig. 8: Example of a two trajectory security graph. Round vertices are checkpoint generated through the heuristic search and triangle vertices are added with the cross-trajectory edges. Additional virtual source v^+ and sink v^- is used later in planning problem.

as $E = E_q \cup \{(v_p^e, v_-)\}_p \cup \{(v_+, v_p^0)\}_p$, where directed edges are added to E from v^+ to all the start vertices, and from all end vertices to v^-, with v_p^0 and v_p^e representing the start and end vertices of sub-team $\mathcal{I}_p$.

Assume that all the robots starting from the virtual source v^+ and end in the virtual sink v^-. Each robot may move from vertex v_i to v_j if $(v_i, v_j) \in E_q$. The path of a robot k can be represented as a flow vector $\mathbf{f}^k = \{f_{ij}^k\}$, where $f_{ij}^k \in \{1, 0\}$ is an indicator variable representing whether robot k's path contains the edge (v_i, v_j). As required by the security guarantee, reference robots must be seen by at least one surveillance robot at every checkpoint. Thus, the planning problem can then be formulated as a path cover problem on G_q, i.e., as finding a set of paths $F = [\mathbf{f}_1, \ldots, \mathbf{f}_\mathcal{K}]$ for cross-trajectory robots such that every checkpoint in V_p is included in at least one path in F. Additionally, to encourage the robots' exchange between different sub-teams, cross-trajectory co-observations are preferred compared to co-observation within the same team. This is achieved with the weights for edges $(v_i, v_j) \in E$ defined as:

$$w_{i,j} = \begin{cases} -w_t & \mathcal{I}_{v_i} = \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ w_c & \mathcal{I}_{v_i} \neq \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ 0 & (v_i, v_j) \in E / E_q \end{cases} \tag{61}$$

where $w_c > w_t$.

With the formulation above, the planning problem can be written as an optimization problem, where we want to balance between the co-observation performance and total number of flows (cross-trajectory robots) needed. For convenience, we use (ij) to represent the edge (v_i, v_j), and $(+i)$ to represent the edge (v^+, v_i).

$$\min \quad \sum_k^\mathcal{K} \sum_{(+i)\in E} f_{+i}^k - \rho \sum_k^\mathcal{K} \sum_{(ij)\in E} w_{ij} f_{ij}^k \tag{62a}$$

$$\text{subject to} \quad \sum_{\{h:(hi)\in E\}} f_{hi}^k = \sum_{\{j:(ij)\in E\}} f_{ij}^k, \qquad \forall k, \forall v \in V_q / \{v^+, v^-\} \tag{62b}$$

$$\sum_k^\mathcal{K} \sum_{\{i:(ij)\in E\}} f_{ij}^k \geq 1 \qquad \forall v_j \in \{V_p\} \tag{62c}$$

$$f_{ij}^k \in \{0, 1\} \qquad \forall (ij) \in E \tag{62d}$$

The first term in 67a is total outgoing flow from the source v^+ while the second term is the total cost of all flows which represents the overall co-observation performance (defined as the total number of cross-trajectory edges taken by all flows beyond the regular trajectory edges). The constant ρ is a penalty parameter manually elected to balance between two terms in the cost function. The constraint (67b) is the flow conservation constraint, which ensures that the amount of flow entering and leaving a given node v is equal (except for v^+ and v^-). The constratin (67c) is the flow coverage constraints, which ensures that all security checkpoints $\{V_p^s\}$ have been visited by at least one flow (robot). Since the security graph G_q is acyclic, this problem is in complexity class P, and can be solved in polynominal time [**?**].

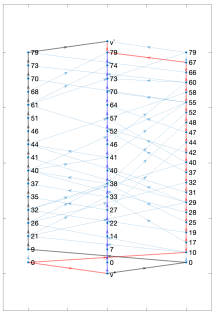


Fig. 9: Security Graph and result for 3 flows

VII. Co-observation performance

Notice that problem (67) is guaranteed to have a solution for $\mathcal{K} = N_p$ where all $\mathcal{K}$ robots follows the reference trajectory ($f_{ij}^k = 1, \forall \mathcal{I}_{v_i} = \mathcal{I}_{v_j} = k$). Additionally, for a fixed number of robots $\mathcal{K} \geq N_p$, it is possible to have a subset of flows $F_\mathcal{E} \in F$ that is empty, i.e. $f_{ij} = 0, \forall (v_i, v_j) \in E$, $f \in F_\mathcal{E}$, these flow will not increase the cost and will not have any impact on the solution. The trade-off between the number of surveillance robots used, and the overall security performance is a critical consideration in this case. Increasing the number of robots generally contributes to improved security by enabling more cross-trajectory co-observations. However, adding more robots can lead to diminishing returns, as the benefits gained from additional robots may be counterbalanced by increased complexity and coordination challenges. To capture this trade-off, the penalty parameter ρ in the cost function 67a allows us to fine-tune the balance between the number of robots employed and the desired task performance. By optimizing this parameter, we can strike a suitable equilibrium between security enhancement and operational efficiency.

To find such tradeoff, we employ an iterative approach. We start with $\mathcal{K} = 1$ and gradually increase it until F contains a empty flow. This indicates that the performance can no longer be improved by adding more robots. To ensure that the value of $\mathcal{K}$ does not grow unbounded, penalty parameter ρ need to ensure that the second term for a single flow in (67a) $\rho \sum_{(ij)\in E} w_{ij} f_{ij}^k$ is always smaller than 1.

VIII. Result and simulation

In this section, we test the proposed method starting from the results in Figure **??**, where three trajectories are provided for the map exploration task with no security related constraints (co-observation schedule and reachability). We use the same environment and dynamic setup, where we have a $10m \times 10m$ task space, two forbidden regions (two rectangles in **??**) and robots with a max velocity of $0.5m/dt$.

We start by adding a total of $\mathcal{K} = 3$ surveillance robots into the workspace using the parameters $w_c = 10$, $w_t = 1$ and $\rho = 0.01$. The three trajectories are transformed into

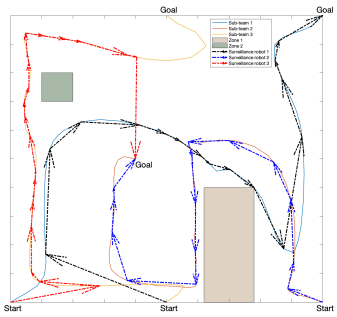


Fig. 10: Result of 3 surveillance agents' plan in workspace

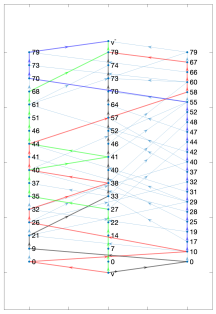


Fig. 11: Security Graph and result for 4 flows.

the graph G shown in Figure 9, where the number on each vertex v_i represents the corresponding time t_i. Vertices in each column belong to the same trajectory, and edges across different columns are cross-trajectory edges.

The flows derived from the solution of the optimization problem (67) are highlighted in the graph. The planning result in the workspace is shown in 10 as dash-dotted arrows with the same color used for each flow in 9. Notice that flows represent the robot co-observation plan instead of actual robots, such that the red flow, for example, shows that the sub-team 2 is expecting a robot from sub-team 1, departing at $t = 0$ and arriving at $t = 10$, and needs to send a robot (not necessarily the same one received from sub-team 1) to sub-team 3, departing at $t = 67$ and arriving at $t = 79$. This plan is not ideal in terms of our security criteria since co-observations happens between the same pair of surveillance robot and sub-team for the majority of the time with only three cross-trajectory edges that are covered.

For the case where $\mathcal{K} = 4$, the results are shown in Figure 11-12. With one additional robot added, there are significant increase in the total cross-trajectory edges covered, and sub-teams performs co-observation with different robots during the task period.

If we further increase $\mathcal{K} > 4$, we do not get a better result; instead, the planner will generate 4 flows with the rest $\mathcal{K} - 4$ flows empty.

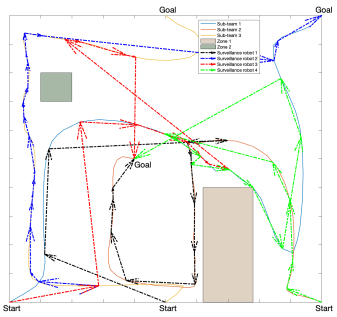


Fig. 12: Result of 3 surveillance agents' plan in work space.

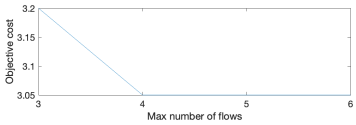


Fig. 13: Plan reach optimality when $\mathcal{K} = 4$, further increase of $\mathcal{K}$ does not increase the cost of objectives.

IX. Summary

In this chapter, we provide a method to enhance the security of a multi-robot system without sacrificing the performance. This is done by introducing additional robots to perform cross-trajectory co-observations by traveling between different trajectories. We model the unsecured multi-robot trajectories as a checkpoint graph by identifying checkpoints that requires observation and cross-trajectory paths that can safely access the checkpoints from different trajectories. We have shown that the co-observation planning problem across different trajectories can be transformed into a multi flow problem on the graph, and that we can find the minimal number of robot needed to finish the co-observation task.

Appendix

A. *Proof of proposition 1*

Proof. For subclaim *1)*a:

$$H^{\mathrm{T}}H = H^2 = 4uu^{\mathrm{T}}uu^{\mathrm{T}} - 4uu^{\mathrm{T}} + I^2 = I, \qquad (63)$$

since $u^{\mathrm{T}}u = 1$.

For subclaim *1)*b, let $U = \begin{bmatrix} u & u_1^{\perp}, u_2^{\perp} \end{bmatrix}$, where $u_1^{\perp}, u_2^{\perp}$ are two orthonormal vectors such that $I = UU^{\mathrm{T}} = uu^{\mathrm{T}} + u_1^{\perp}(u_1^{\perp})^{\mathrm{T}} + u_2^{\perp}(u_2^{\perp})^{\mathrm{T}}$; then, substituting I in (16), we have that the eigenvalue decomposition of H is given by

$$H = U\,\mathrm{diag}(1, -1, -1)U^{\mathrm{T}}. \qquad (64)$$

Since the determinant of a matrix is equal to the product of the eigenvalues, $\det(H) = 1$.

For subclaim *2)*, first note that $Hu = 2uu^{\mathrm{T}}u - u = u$. It follows that the sum of $\nu_{\mathcal{F}}$ and $\nu_{\mathcal{E}}$ is invariant under H:

$$H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) = Hu\|\nu_{\mathcal{F}} + \nu_{\mathcal{E}}\| = u\|\nu_{\mathcal{F}} + \nu_{\mathcal{E}}\| = \nu_{\mathcal{F}} + \nu_{\mathcal{E}}, \quad (65)$$

and that their difference is flipped under H:

$$H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) = 2uu^{\mathrm{T}}(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) - (\nu_{\mathcal{F}} - \nu_{\mathcal{E}})^2 = -(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}). \tag{66}$$

Combining (70) and (71) we obtain

$$H\nu_{\mathcal{F}} = \frac{1}{2}\big(H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) + H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}})\big) = \nu_{\mathcal{E}} \tag{67}$$

□

B. Proof or proposition 2

Proof. From the definition of H in (16), we have

$$\dot{H} = 2(\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) \tag{68}$$

Recall that $\dot{u} = \frac{1}{\|u'\|}(I - uu^{\mathrm{T}})\dot{u}'$ (see, for instance, [**?**]), which implies $(I - uu^{\mathrm{T}})\dot{u}' = \dot{u}'$. It follows that $\dot{u}$ flips sign under the action of H^{T}:

$$\begin{aligned} H^{\mathrm{T}}\dot{u} &= (2uu^{\mathrm{T}} - I)\frac{\big(I - uu^{\mathrm{T}}\big)}{\|u'\|}\dot{u}' \\ &= \frac{1}{\|u'\|}(2uu^{\mathrm{T}} - I - 2uu^{\mathrm{T}}uu^{\mathrm{T}} + uu^{\mathrm{T}})\dot{u}' \\ &= -\frac{1}{\|u'\|}(I - uu^{\mathrm{T}})\dot{u}' = -\dot{u} \end{aligned} \tag{69}$$

Inserting $HH^{\mathrm{T}} = I$ in (73), and using 18, we finally have

$$\begin{aligned} \dot{H} =& 2HH^{\mathrm{T}}(\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) = 2H(-\dot{u}u^{\mathrm{T}} + u\dot{u}^{\mathrm{T}}) \\ =& -2H[[u]_\times \dot{u}]_\times \\ =& -2H\left[[u]_\times \frac{\big(I - uu^{\mathrm{T}}\big)\big(I - \nu_{\mathcal{F}}\nu_{\mathcal{F}}^{\mathrm{T}}\big)}{\|u'\|\,\|\nu_{\mathcal{F}}\|}\dot{\nu}_{\mathcal{F}}\right]_\times \\ =& -2H[M\dot{\nu}_{\mathcal{F}}]_\times, \end{aligned} \tag{70}$$

which is equivalent to the claim. □

C. Proof of proposition 3

Proof. To make the notation more compact, we will use $\partial_q f$ instead of $\partial_{\left[\begin{smallmatrix} q_1 \\ q_2 \end{smallmatrix}\right]} f$ for the remainder of the proof. The differential of (36) can be represented as:

$$\begin{aligned} \dot{\pi}_{p\mathcal{E}} =& \dot{H}^{-1}SH(q_{avoid} - o) + H^{-1}\dot{S}H(q_{avoid} - o) \\ &+ H^{-1}S\dot{H}(q_{avoid} - o) + (H^{-1}SH - I)\dot{o} \end{aligned} \tag{71}$$

where

$$\begin{aligned} \dot{S} =& -S^2(Q\dot{s} + s\dot{Q}) \\ =& -S^2(Q\partial_q s\dot{q} - \partial_b Q\partial_q b\dot{q}) \end{aligned} \tag{72}$$

where

$$\partial_b Q = 2\frac{s}{b^3}\,\mathrm{diag}\{0, 1, 1\} \tag{73}$$

To compute the derivative $\partial_q \pi$, we need the expression of $\partial_q b$, $\partial_q o$ and $\partial_q s$; the first two can be easily derived using the equations above:

$$\partial_q b = \frac{1}{4b}\left[q_1 - q_2, q_2 - q_1\right]^{\mathrm{T}} \tag{74}$$

$$\partial_q o = \big[I/2, I/2\big]^{\mathrm{T}} \tag{75}$$

In order to get $\partial_q s$, we use the fact that $F\big(s(q)\big) = 0$ for all q; hence $F\big(\tilde{q}(t)\big) \equiv 0$, and $\partial_q F = 0$. We then have:

$$0 = \dot{F} = 2q^{\mathrm{T}}Q'\dot{q} + q^{\mathrm{T}}\partial_s Q' q\dot{s} + q^{\mathrm{T}}\partial_b Q' q\dot{b} \tag{76}$$

where

$$\partial_s Q' = -\,\mathrm{diag}\left(\frac{2a^2}{(s+a^2)^3}, \frac{2b^2}{(s+b^2)^3}, \frac{2b^2}{(s+b^2)^3}\right). \tag{77}$$

By moving term $\dot{s}$ to the left-hand side we can obtain:

$$\begin{aligned} \dot{s} &= (q^{\mathrm{T}}\partial_s Q' q)^{-1}(2q^{\mathrm{T}}Q'\dot{q} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \\ &= (q^{\mathrm{T}}\partial_s Q' q)^{-1}(-4q^{\mathrm{T}}Q'H[U\dot{q}]_\times(q_{avoid} - o) \\ &\qquad - 2q^{\mathrm{T}}Q'H\dot{o} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \\ &= (q^{\mathrm{T}}\partial_s Q' q)^{-1}(-4q^{\mathrm{T}}Q'H[q_{avoid} - o]_\times U\dot{q} \\ &\qquad - 2q^{\mathrm{T}}Q'H\dot{o} + q^{\mathrm{T}}\partial_b Q' q\dot{b}) \end{aligned} \tag{78}$$

The second term of equation (76) turns into:

$$\begin{aligned} H^{-1}\dot{S}H(q_{avoid} - o) =& -H^{-1}Q'q\dot{s} - sH^{-1}S^2\partial_b Q q\dot{b} \\ =& \big((q^{\mathrm{T}}\partial_s Q' q)^{-1}H^{-1}Q'qq^{\mathrm{T}}(4Q'H[q_{avoid} - o]_\times U \\ &+ 2Q'H\partial_q o - \partial_b Q' qq\partial_q b) - sH^{-1}S^2\partial_b Q q\partial_q b\big)\dot{q} \end{aligned} \tag{79}$$

Thus equation (76) could be written as:

$$\begin{aligned} \dot{\pi}_{p\mathcal{E}} =& \big(-2H[SH(q_{avoid} - o)]_\times U \\ &+ \big((q^{\mathrm{T}}\partial_s Q' q)^{-1}H^{-1}Q'qq^{\mathrm{T}}(4Q'H[q_{avoid} - o]_\times U \\ &+ 2Q'H\partial_q o - \partial_b Q' qq\partial_q b) - sH^{-1}S^2\partial_b Q q\partial_q b\big) \\ &- 2H^{-1}SH[q_{avoid} - o]_\times U \\ &+ (H^{-1}SH - I)\partial_q o\big)\dot{q}, \end{aligned} \tag{80}$$

from which the claim follows. □

D. Proof of proposition 4

Proof. We first need to derive $\dot{d}_{\mathcal{E}}$ and $\dot{d}_{\mathcal{E}t}$

$$\dot{d}_{\mathcal{E}} = -n^{\mathrm{T}}\partial_q o\dot{q} \tag{81}$$

$$\begin{aligned} \dot{d}_{\mathcal{E}t} =& (\dot{n}_{\mathcal{E}}^{\mathrm{T}}Q^{-1}n_{\mathcal{E}} + n_{\mathcal{E}}^{\mathrm{T}}\dot{Q}^{-1}n_{\mathcal{E}} + n_{\mathcal{E}}^{\mathrm{T}}Q^{-1}\dot{n}_{\mathcal{E}})/\sqrt{n_{\mathcal{E}}^{\mathrm{T}}Q^{-1}n_{\mathcal{E}}} \\ =& (\sqrt{n_{\mathcal{E}}^{\mathrm{T}}Q^{-1}n_{\mathcal{E}}})^{-1}\left(-2n^{\mathrm{T}}H[Q^{-1}n_{\mathcal{E}}]_\times U\right. \\ &\left. + n_{\mathcal{E}}^{\mathrm{T}}\partial_b Q^{-1}n_{\mathcal{E}}\partial_q b - 2n_{\mathcal{E}}Q^{-1}H[n]_\times U\right)\dot{q} \end{aligned} \tag{82}$$

Next, we need to derive $\dot{p}_{t1}$, $\dot{p}_{t2}$ and $\dot{p}_{\mathcal{L}}$. Since $p_{\mathcal{L}}$ could be written as

$$p_{\mathcal{L}} = \frac{d_{\mathcal{E}}Q^{-1}n_{\mathcal{E}}}{d_{\mathcal{E}t}^2}, \tag{83}$$

we have

$$\begin{aligned} \dot{p}_{\mathcal{L}} =& \left((-\frac{d_{\mathcal{E}t}n^{\mathrm{T}}\partial_q o - 2d_{\mathcal{E}}\partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^3})Q^{-1}n_{\mathcal{E}}\right. \\ &\left. + \frac{d_{\mathcal{E}}\partial_b Q^{-1}n_{\mathcal{E}}\partial_q b - 2d_{\mathcal{E}}Q^{-1}H[n]_\times U}{d_{\mathcal{E}t}^2}\right)\dot{q} \end{aligned} \tag{84}$$

$$\begin{aligned} \dot{p}_1 =& \left(-\frac{Q^{-1}n_{\mathcal{E}}\partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^2}\right. \\ &\left. + \frac{\partial_b Q^{-1}n_{\mathcal{E}}\partial_q b - 2Q^{-1}H[n]_\times U}{d_{\mathcal{E}t}}\right)\dot{q} \end{aligned} \tag{85}$$

subtracting $\dot{q}$ from (56) and (57), we can derive the result shown in (45) □

REFERENCES

[1] Gonzalo Pajares. Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs). *Photogrammetric Engineering & Remote Sensing*, 81(4):281–330, 2015.

[2] Brian J Julian, Michael Angermann, Mac Schwager, and Daniela Rus. Distributed robotic sensor networks: An information-theoretic approach. *The International Journal of Robotics Research*, 31(10):1134–1154, 2012.

[3] Martin Brunner, Hans Hofinger, Christoph Krauß, Christopher Roblee, P Schoo, and S Todt. Infiltrating critical infrastructures with next-generation attacks. *Fraunhofer Institute for Secure Information Technology (SIT), Munich*, 2010.

[4] Conner Forrest. Robot kills worker on assembly line, raising concerns about human-robot collaboration. https://tinyurl.com/y2tzg4te, March 2017.

[5] Kacper Wardega, Roberto Tron, and Wenchao Li. Resilience of multi-robot systems to physical masquerade attacks. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 120–125. IEEE, 2019.

[6] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[7] Ziqi Yang and Roberto Tron. Multi-agent path planning under observation schedule constraints. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6990–6997. IEEE, 2020.

[8] Alston S Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342, 1958.

[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. 3(1):1–122, 2011.

[10] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.