

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 feasibility problems of the first one, and uses the same reachability 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

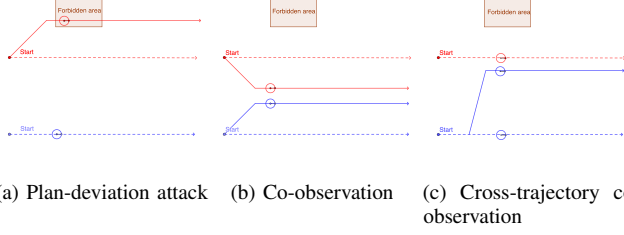


Fig. 1: 1b Limited by the co-observation requirement, both red and blue robot follow the co-observation secured routes (solid lines) and abandon the optimal ones (dashed line). 1c Through cross-trajectory co-observations, the blue team sends one robot to follow the red team (solid blue line) and performs 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 paper, 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. MULTI-ROBOT TRAJECTORY PLANNING WITH CO-OBSERVATION

In this section, we aim to enhance the security of MRSs against malicious takeovers and deviations by attackers. Specifically, we design an inter-robot observation plan (co-observation schedule) to ensure that, during the task period, robots are constantly in proximity to one another according to a schedule, in order to observe and detect potential hazardous behaviors. This is achieved by keeping the potential region that the robots can possibly reach between each consecutive co-observations away from the forbidden regions. In this way, the plan ensures that any potential deviations to these forbidden regions will cause the corresponding robot to miss their next co-observation with other robots. [5] solved this problem in grid world for robots with fixed start and goal locations (Figure 2b) but does not offer the ability to optimize arbitrary smooth cost functions of the trajectories.

We formulate the planning problem as an optimal trajectory optimization problem to minimize arbitrary smooth objective functions. We denote as $q_{ij} \in \mathbb{R}^m$ the position of agent i at the discrete-time index j , with m representing the dimension of the state space. For a team of n agents, and a task time horizon of T , the overall trajectory of the multi-agent system can be represented as an aggregated vector $\mathbf{q} \in \mathbb{R}^{nmT}$. The goal of our path planning problem is to minimize or maximize an objective function $\Phi(\mathbf{q})$ under a set of nonlinear constraints described by a set Ω . The set Ω is given by the intersection of spatio-temporal sets given by the security constraints (co-observation schedule, reachability analysis) and the traditional path planning constraints. Formally:

$$\begin{aligned} & \min / \max \quad \Phi(\mathbf{q}) \\ & \text{subject to} \quad \mathbf{q} \in \Omega. \end{aligned} \quad (1)$$

To give a concrete example of the cost Φ and the set Ω , we introduce a representative application that will be used for all the simulations throughout the paper.

Example 1. We consider the estimation of a slowly-varying scalar or vector field, denoted as \mathbf{x} , at discrete locations within an unknown environment. Autonomous agents navigate this environment, collecting sensory data to construct a corresponding map (see Figure 2a). Our map comprises points of interest arranged on a grid, each associated with a slowly-changing value. Our goal is to find paths that minimize uncertainty and effectively reconstruct the field. We utilize Kalman Filters (KFs) (cf. [?]) to estimate uncertainty through the covariance P_j . Additionally, the robot provides field measurements at waypoints q using a Gaussian radial basis function for radius and quality modeling.

Our cost function $\Phi(\mathbf{q})$ measures the maximum uncertainty P_j among map locations of interest along the entire trajectory \mathbf{q} , and our objective is to minimize this maximum uncertainty. Constraints Ω includes traditional constraints like velocity constraints, convex obstacles, waypoints to reach with deadlines, and security constraints including co-observation constraints and reachability constraints.

This section is organized as follows. We begin by presenting the foundational mathematical concepts, notably the basic

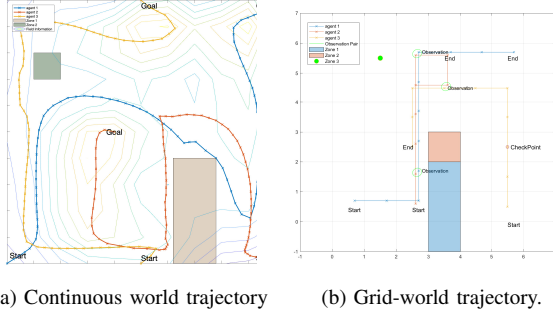


Fig. 2: Trajectory design of an example map exploration task for a three-robot system with start locations and destinations fixed. Task are planned in a 8×8 grid world. Zone 1 is obstacle, Zone 2 and Zone 3 are safe zones. Fig 2b shows the planned trajectory with co-observation schedule in grid-world. Fig 2a shows the trajectory design in continuous world optimized with respect to a map exploration task

ADMM optimal trajectory solver. Subsequently, our solver allows us to seamlessly integrate security constraints, encompassing co-observation schedule and reachability constraints. Concluding this section, we present the obtained results alongside limitations (due to infeasibility) that are addressed in ??.

A. Preliminaries

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

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

$$\left. \frac{d}{dt} f(x(t)) \right|_{t=0} = \partial_x f(x(0)) \dot{x}(0) \quad (2)$$

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 brevity we will use \dot{f} for $\frac{d}{dt} f$ and $\partial_x f$ for $\frac{\partial f}{\partial x}$. The differentials $\partial_x f$ is derived through (2) 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 $f : \mathbb{R}^R \rightarrow \mathbb{R}^{m \times n}$. Note that in this case (2) is still formally correct, although the RHS needs to interpret $\partial_x f$ as a linear operator applied to \dot{x} .

2) *Householder rotations*: In the derivation of the reachability constraint in ??, we need to define a differentiable transformation 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

$$u' = \nu_{\mathcal{F}} + \nu_{\mathcal{E}}, \quad u = \frac{u'}{\|u'\|}. \quad (3)$$

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

$$H(\nu_{\mathcal{F}}, \nu_{\mathcal{E}}) = 2uu^T - I. \quad (4)$$

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) $H^T H = I$;
 - b) $\det(H) = 1$.
- 2) $\nu_{\mathcal{E}} = H\nu_{\mathcal{F}}$.

Proof. See Appendix A. \square

We compute the differential of H implicitly using the relation (2). We will use the notation $[v]_{\times} : \mathbb{R}^3 \rightarrow \mathbb{R}^{3 \times 3}$ to denote the matrix representation of the cross product with the vector v , i.e.,

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \quad (5)$$

such that $[v]_{\times} w = v \times w$ for any $w \in \mathbb{R}^3$.

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

$$\dot{H} = H[-2M\dot{\nu}_{\mathcal{F}}]_{\times} \quad (6)$$

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

$$M = [u]_{\times} \frac{(I - uu^T)(I - \nu_{\mathcal{F}}\nu_{\mathcal{F}}^T)}{\|u'\|\|\nu_{\mathcal{F}}\|}. \quad (7)$$

Proof. See Appendix B. \square

3) *Alternating Directions Method of Multipliers (ADMM)*: The basic idea behind our ADMM-based solver [?] is to split the constraints from the objective function using a different set of variables \mathbf{z} , and then solve an Augmented Lagrangian formulation of the optimization problem in (1). More specifically, we can rewrite the constraint $\mathbf{q} \in \Omega$ using an indicator function Θ , and include it in the objective function. In the traditional application of ADMM, the variables \mathbf{z} are duplicates of \mathbf{x} projected to the constraint set Ω . However, in path planning problems, some constraints are non-convex, rendering the projection step more difficult (due to the presence of multiple local minima). To deal with this problem, we propose a minor generalization of the ADMM formulation where we allow \mathbf{z} to replicate an arbitrary function of the main variables \mathbf{q} (instead of being an exact copy):

$$\begin{aligned} \max \quad & \Phi(\mathbf{q}) + \Theta(\mathbf{z}) \\ \text{s.t.} \quad & D(\mathbf{q}) - \mathbf{z} = 0 \end{aligned} \quad (8)$$

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

The update steps of the ADMM algorithm can be then denoted as [9]:

$$\mathbf{q}^{k+1} := \underset{\mathbf{q}}{\operatorname{argmin}} (\Phi(\mathbf{q}^k) + \frac{\rho}{2} \|D(\mathbf{q}) - \mathbf{z}^k + \mathbf{u}^k\|_2^2) \quad (9a)$$

$$\mathbf{z}^{k+1} := \Pi_{\mathcal{Z}}(D(\mathbf{q}^{k+1}) + \mathbf{u}^k) \quad (9b)$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}, \quad (9c)$$

where $\Pi_{\mathcal{Z}}$ is the new projection to the modified constraint set \mathcal{Z} . **It is not clear where \mathcal{Z} comes from**, \mathbf{u} represents a scaled dual variable that, intuitively, accumulates the sum of primal residuals

$$\mathbf{r}^k = D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}. \quad (10)$$

Checking the primal residuals alongside with the dual residuals

$$\mathbf{s}^k = -\rho(\mathbf{z}^k - \mathbf{z}^{k-1}), \quad (11)$$

after each iteration, the steps are reiterated until convergence when the primal and dual residuals are small, or divergence when primal and dual residual remains large after a fixed large number of iterations. Since $D(\mathbf{q})$ is the vertical concatenation of $D_i(\mathbf{q})$, the projection of each set \mathcal{Z}_i is independent for each constraint and can be computed separately. The advantage of this formulation is that we can choose $D(\mathbf{q})$ such that the new constraint set \mathcal{Z} 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{q} in (9a). Additionally, the function $D(\mathbf{q})$ can be used to select only the subset of the variables on which a constraint depends, speeding up computations.

We now provide the functions $D(\mathbf{q})$, the sets \mathcal{Z} , and the corresponding projection operators $\Pi_{\mathcal{Z}}$ for traditional path planning constraints (Section II-B-Section II-D), co-observation security constraints (Section II-E), and reachability constraints (??). The latter are based on the definition of *ellipsoidal reachability regions* (Section II-F).

B. 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 $D(\mathbf{q})$ to return the velocity vectors for the i -th agent at time step j ; its result then needs to be constrained to a sphere.

ADMM constraint 1 (Velocity constraint).

$$D_{ij}(\mathbf{q}) = q_{ij} - q_{i(j-1)}, \quad j \in \{1, \dots, T\} \quad (12)$$

$$\mathcal{Z}_{ij} = \{z \mid \|z\| \leq v_{max}\} \quad (13)$$

$$\Pi_{\mathcal{Z}_{ij}}(z) = \begin{cases} v_{max} \frac{z}{\|z\|} & \text{if } \|z\| > v_{max} \\ z & \text{otherwise} \end{cases} \quad (14)$$

C. Convex polygonal obstacles collision constraints

We use convex polygons to model regions defining solid obstacles and forbidden regions that cannot be entered by agents. The convex region \mathcal{P} is defined using a collection of l hyperplanes $n_k q = m_k$, where n_k is the normal vector pointing toward the outside of the obstacle and m_k is the scalar offset

defining the k -th hyperplane. The distance between a single waypoint q_{ij} and the k th hyperplane $d_k(q_{ij}, \mathcal{P})$ is defined as:

$$d_k(q_{ij}, \mathcal{P}) = p_{ij}^T n_k - m_k. \quad (15)$$

We then defined function $D(\mathbf{q})$ to return the maximum distance from q_{ij} to all hyperplane boundaries of region \mathcal{P} , and set constraint on z that it is a non-negative scalar. This constraint is applied to all $q_{ij} \in \mathbf{q}$.

ADMM constraint 2 (Obstacle constraint).

$$D(q_{ij}) = \max_{k=1, \dots, l} (d_k(q_{ij}, \mathcal{P})) \quad (16)$$

$$\mathcal{Z} = \{z \mid z \geq 0\}, \quad (17)$$

$$\Pi_{\mathcal{Z}}(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases} \quad (18)$$

The motivating idea for (17) and (18) is to identify waypoints within a specific region and project them to the closest boundary. Non-convex obstacles can be handled by the union of (possibly overlapping) convex obstacles. By setting the normal vector pointing toward the inside of the region, this method can also serve to constrain the robots within the designated workspace region.

D. Waypoints with flexible deadlines

It is common for security related constraints to require robot reach a specific location at specific time. We assume robot i is required to reach a given point p within a radius of d_{max} at some time instant j belonging to a given time window $[t_1, t_2]$. The constraint function $D(\mathbf{q})$ to return the smallest distance from the point p to waypoint on the trajectory restricted to the relevant time window.

ADMM constraint 3 (Waypoint constraint).

$$D(\mathbf{q}) = \min_{j \in \{t_1, \dots, t_2-1\}} (\operatorname{dist}(p, \overrightarrow{q_{ij}q_{i(j+1)}})) \quad (19)$$

$$\mathcal{Z} = \{z \mid z < z_{max}\}, \quad (20)$$

$$\Pi_{\mathcal{Z}}(z) = \min(z, d_{max}). \quad (21)$$

where $\operatorname{dist}(p, \overrightarrow{q_{ij}q_{i(j+1)}})$ 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. **Give example application here, and include application of ADMM without the security constraints, but highlighting a path that would break the security** **zyang** plot new simpler example, add collision avoidance

E. Co-observation schedule constraint

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. This constraint is modeled as a relative distance constraint between two robots at some time instant j (i.e., to require two

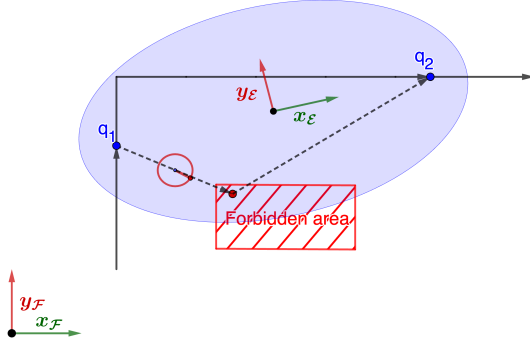


Fig. 3: The ellipse showcases the reachability region. The black line is the planned trajectory of a robot, q_1 and q_2 are two locations this robot are expected at given time t_1 and t_2 , dashed line is a possible trajectory if the robot is compromised after reaching q_1 . Axis of global frame \mathcal{F} and canonical frame $\mathcal{F}_\mathcal{E}$ are shown as $x_\mathcal{F}, y_\mathcal{F}$ and $x_\mathcal{E}, y_\mathcal{E}$ respectively.

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 II-F, has an empty intersection with all forbidden regions. We write the function for the constraint as:

ADMM constraint 4 (Co-observation constraint).

$$D(\mathbf{q}) = \overrightarrow{q_{aj}q_{bj}}, \quad (22)$$

$$\mathcal{Z} = \{z \mid z \leq d_{max}\}, \quad (23)$$

$$\Pi_{\mathcal{Z}}(z) = \begin{cases} d_{max} \frac{z}{\|z\|} & \text{if } \|z\| > d_{max}, \\ z & \text{otherwise.} \end{cases} \quad (24)$$

where a, b are the indices of the pair of agents required for a mutual inspection. The locations q_{aj} and q_{bj} where the co-observation performed are computed as part of the optimization.

F. Definition of ellipsoidal reachability regions

In this section, we define *ellipsoidal reachability regions* based on pairs of locations on a trajectory, and a transformation of such region in axis-aligned form. These definitions will be used to define different types of reachability constraints (between an ellipsoid and different geometric entities such as a point, line, line segment, and polygon), their projections, and differentials in subsequent sections. These will be used in the ADMM formulation in section II-A3.

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 $a = \frac{v_{max}}{2}(t_2 - t_1)$.

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.

1) *Transformation to canonical coordinates:* To simplify the problem, we employ a canonical rigid body transformation that repositions the ellipse \mathcal{E} from the global frame \mathcal{F} to a canonical frame $\mathcal{F}_\mathcal{E}$. In this new frame, the ellipsoid's center aligns with the origin, and its foci align with the first axis of $\mathcal{F}_\mathcal{E}$. We perform this transformation in a differentiable manner, with respect to the two foci serving as waypoints for a robot.

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, 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:

$$q^\mathcal{F} = Rq^\mathcal{E} + o, \quad q^\mathcal{E} = R^\mathsf{T}(q^\mathcal{F} - o). \quad (25)$$

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:

$$\nu'_\mathcal{F} = q_2 - q_1, \quad \nu_\mathcal{F} = \frac{\nu'_\mathcal{F}}{\|\nu'_\mathcal{F}\|}, \quad \nu_\mathcal{E} = [1, 0, 0]^\mathsf{T}; \quad (26)$$

see Fig.3 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 (25) we see that o represents the center of $\mathcal{E}(q_1, q_2)$ expressed in \mathcal{F} , i.e.:

$$R = H(\nu_\mathcal{F}(q_1, q_2), \nu_\mathcal{E}(q_1, q_2)), \quad o = \frac{1}{2}(q_1 + q_2). \quad (27)$$

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

$$q_1^\mathcal{E} = [c \ 0 \ 0]^\mathsf{T}, \quad q_2^\mathcal{E} = [-c \ 0 \ 0]^\mathsf{T}, \quad (28)$$

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

Definition 4. The ellipsoid \mathcal{E} in the canonical frame can be described as the zero level set of the quadratic function

$$E^\mathcal{E}(q^\mathcal{E}) = q^\mathcal{E}^\mathsf{T} Q q^\mathcal{E} - 1 \quad (29)$$

where

$$Q = \text{diag}(a^{-2}, b^{-2}, b^{-2}), \quad (30)$$

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

For all reachability ellipsoid constraints, we first solve the problem in the canonical frame $\mathcal{F}_\mathcal{E}$, and transform the solutions to the global frame \mathcal{F} using the defined transformation function.

We are now ready to formulate constraints based on reachability ellipsoids 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 $D(q)$, its differential $\partial_q D(q)$, the set ζ and the projection Π_ζ that can be incorporated in the ADMM optimization (8).

G. Point-ellipsoid reachability constraint

As shown in Fig.4, 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 $q_{avoid} \notin \mathcal{E}(q_1, q_2)$. This constraint can be written as,

ADMM constraint 5 (Point-ellipsoid reachability constraint).

$$D(q) = \begin{cases} \pi_{p\mathcal{E}}(q) - q_{avoid} & q_{avoid} \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \quad (31)$$

$$\mathcal{Z} = \{q \in \mathbb{R}^{nm} : \|D_p(q)\| = 0\}, \quad (32)$$

$$\Pi_{\mathcal{Z}}(z) = 0, \quad (33)$$

where $\pi_{p\mathcal{E}}(q_{avoid}; q_1, q_2, a) = q_p$ is a projection function that returns a projected point q_p of q_{avoid} outside the ellipse, i.e., as the solution to

$$\pi_{p\mathcal{E}} = \underset{q_p \in \mathcal{E}^c}{\operatorname{argmin}} \|q_{avoid} - q_p\|^2 \quad (34)$$

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

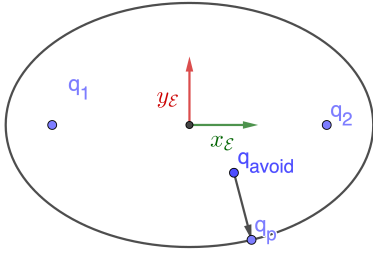


Fig. 4: Point-ellipsoid constraint, a point q_{avoid} inside ellipsoid is projected to the areas outside the ellipsoid \mathcal{E} .

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.

zyang can this be treated as proposition

Proposition 3. In canonical frame $\mathcal{F}_{\mathcal{E}}$, the projection from a point $q_{avoid} \in \mathcal{E}$ inside the ellipse to the boundary of \mathcal{E} :

$$q_p^{\mathcal{E}} = (I + sQ)^{-1} q_{avoid}^{\mathcal{E}} = S q_{avoid}^{\mathcal{E}} \quad (35)$$

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 (36) in $E^{\mathcal{E}}(q^{\mathcal{E}})$:

$$0 = F(s) = q_p^{\mathcal{E}T} Q q_p^{\mathcal{E}} - 1 = q_{avoid}^{\mathcal{E}T} Q'(s) q_{avoid}^{\mathcal{E}} - 1, \quad (36)$$

where

$$Q'(s) = S^T Q S = \operatorname{diag} \left(\frac{a^2}{(s+a^2)^2}, \frac{b^2}{(s+b^2)^2}, \frac{b^2}{(s+b^2)^2} \right) \quad (37)$$

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

1) Projection to the standard ellipse: **rtron** Move to the appendix or refer to the conference paper

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

We now turn our attention to the problem of projecting $q_{avoid}^{\mathcal{E}}$ 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 ellipse, 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 (34) (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 $E^{\mathcal{E}}$, i.e.

$$q_p^{\mathcal{E}} - q_{avoid}^{\mathcal{E}} = s \partial_q E^{\mathcal{E}}(q_p^{\mathcal{E}})^T = s Q q_p^{\mathcal{E}} \quad (38)$$

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

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

$$\begin{aligned} \pi_{p\mathcal{E}}(q) &= R^{-1}(q(t_1), q(t_2)) q_p^{\mathcal{E}} + o \\ &= R^{-1}(q(t_1), q(t_2)) S q_{avoid}^{\mathcal{E}} + o \\ &= R^{-1} S R(q_{avoid} - o) + o \end{aligned} \quad (39)$$

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 $\pi_{p\mathcal{E}}^{2D} = P \pi_{p\mathcal{E}}^{3D}(P^T q_{avoid}; P^T q_1, P^T q_2, a)$.

2) ADMM constraints: The corresponding constraint is written as

$$D_p(q) = \begin{cases} \pi_{p\mathcal{E}}(q) - q_{avoid} & q_{avoid} \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \quad (40)$$

and feasible set and projection function as:

$$\zeta = \{q \in \mathbb{R}^{nm} : \|D_p(q)\| = 0\}, \quad \Pi(D_p(q)) = 0 \quad (41)$$

Proposition 4. 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 $q_{avoid}^{\mathcal{E}}$)

$$\begin{aligned} \partial_{[q_1, q_2]} \pi_{p\mathcal{E}} &= -2H[SH(q-o)]_{\times} U \\ &\quad + ((q^T \partial_s Q' q)^{-1} H^{-1} Q' q q^T (4Q' H[q-o]_{\times} U \\ &\quad + 2Q' H \partial_q o - \partial_b Q' q q \partial_q b) - s H^{-1} S^2 \partial_b Q q \partial_q b) \\ &\quad - 2H^{-1} SH[q-o]_{\times} U + (H^{-1} SH - I) \partial_q o \end{aligned} \quad (42)$$

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

H. Plane-ellipsoid reachability constraint

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 $\mathcal{L}(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^n : \mathbf{n}^T \tilde{q} = d\}$ (as shown in Figure 5). 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 (II-F1), the ellipse can be

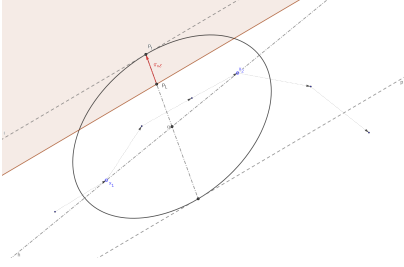


Fig. 5: Plane-ellipse constraint. Projected a ellipsoid to one side of a plane. The projection is simplified to the point-ellipse constraint as projecting a point inside ellipse p_L to p_t on outside the ellipse.

transferred into a standard one with the new hyperplane in the form of $\mathcal{L}^\varepsilon(\tilde{q}) = \{\tilde{q} \in \mathbb{R}^m : \mathbf{n}_\varepsilon^\top \tilde{q} = d_\varepsilon\}$, where,

$$\mathbf{n}_\varepsilon = H(q_1, q_2)\mathbf{n} \quad (43)$$

$$d_\varepsilon = -\mathbf{n}^\top \mathbf{o} + d \quad (44)$$

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

$$d_{\varepsilon t} = \sqrt{\mathbf{n}_\varepsilon^\top Q^{-1} \mathbf{n}_\varepsilon}. \quad (45)$$

The corresponding tangent points are given as

$$p_1^\varepsilon = \frac{d_{\varepsilon t} Q^{-1} \mathbf{n}_\varepsilon}{\mathbf{n}_\varepsilon^\top Q^{-1} \mathbf{n}_\varepsilon} = \frac{Q^{-1} \mathbf{n}_\varepsilon}{d_{\varepsilon t}}, \quad (46)$$

$$p_2^\varepsilon = -p_1^\varepsilon. \quad (47)$$

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

Definition 5. The tangent interpolation point is defined as

$$p_\mathcal{L}^\varepsilon = \frac{d_\varepsilon Q^{-1} \mathbf{n}_\varepsilon}{\mathbf{n}_\varepsilon^\top Q^{-1} \mathbf{n}_\varepsilon} \quad (48)$$

Note that $p_\mathcal{L}^\varepsilon \in \mathcal{L}$ and when $d_\varepsilon = d_{\varepsilon t}$ or $d_\varepsilon = -d_{\varepsilon t}$, $p_\mathcal{L}^\varepsilon = p_1^\varepsilon$ or $p_\mathcal{L}^\varepsilon = p_2^\varepsilon$, respectively. It is clear that when $d_\varepsilon \in [-d_{\varepsilon t}, d_{\varepsilon t}]$, 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:

$$\pi_{\mathbf{n}\varepsilon}^\varepsilon(q) = \begin{cases} p_{t1}^\varepsilon - p_\mathcal{L}^\varepsilon & d_\varepsilon \in [0, d_{\varepsilon t}], \\ p_{t2}^\varepsilon - p_\mathcal{L}^\varepsilon & d_\varepsilon \in [-d_{\varepsilon t}, 0), \\ 0 & \text{otherwise.} \end{cases} \quad (49)$$

2) *ADMM constraints:* Transforming the projection π^ε back to \mathcal{F} we have:

$$D_{\mathbf{n}}(q) = H^{-1}(q(t_1), q(t_2))\pi_{\mathbf{n}\varepsilon}^\varepsilon(q) + o, \quad (50)$$

with the corresponding constraint written as

$$\zeta_{\mathbf{n}} = \{q \in \mathbb{R}^{nm} : \|D_{\mathbf{n}}(q)\| = 0\} \quad (51)$$

$$\Pi_{\mathbf{n}}(D_{\mathbf{n}}(q)) = 0 \quad (52)$$

3) *Gradients for the projection function:*

Proposition 5. The differential of the projection function $\Pi_{\mathbf{n}\varepsilon}^\varepsilon(q)$ with respect to the foci q_1 and q_2 is given by:

$$\partial_q \pi_{\mathbf{n}\varepsilon}^\varepsilon(q) = \begin{cases} \partial_q p_{t1}^\varepsilon - \partial_q p_\mathcal{L}^\varepsilon & d_\varepsilon \in [0, d_{\varepsilon t}], \\ \partial_q p_{t2}^\varepsilon - \partial_q p_\mathcal{L}^\varepsilon & d_\varepsilon \in [-d_{\varepsilon t}, 0), \\ 0 & \text{otherwise.} \end{cases} \quad (53)$$

where

$$\begin{aligned} \partial_q p_\mathcal{L} = & \left(-\frac{d_{\varepsilon t} \mathbf{n}^\top \partial_q \mathbf{o} - 2d_\varepsilon \partial_q d_{\varepsilon t}}{d_{\varepsilon t}^3} \right) Q^{-1} \mathbf{n}_\varepsilon \\ & + \frac{d_\varepsilon \partial_b Q^{-1} \mathbf{n}_\varepsilon \partial_q b - 2d_\varepsilon Q^{-1} H[n] \times U}{d_{\varepsilon t}^2}, \end{aligned} \quad (54)$$

$$\partial_q p_1 = -\frac{Q^{-1} \mathbf{n}_\varepsilon \partial_q d_{\varepsilon t}}{d_{\varepsilon t}^2} + \frac{\partial_b Q^{-1} \mathbf{n}_\varepsilon \partial_q b - 2Q^{-1} H[n] \times U}{d_{\varepsilon t}}. \quad (55)$$

Based on the previous proposition, the differential of (50) can be written as:

$$\partial_q D_{\mathbf{n}\varepsilon} = -2H[\Pi_{\mathbf{n}\varepsilon}^\varepsilon] \times M + H^{-1} \partial_q \Pi_{\mathbf{n}\varepsilon}^\varepsilon \quad (56)$$

I. Line-segment-ellipse reachability constraint

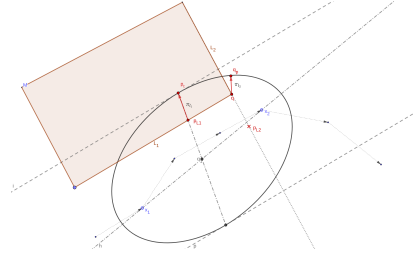


Fig. 6: Polygon-ellipse constraint. This constraint is simplified to either a plane-ellipse constraint or a point-ellipse constraint.

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:

$$\begin{bmatrix} (p_1 - p_2)^\top \\ (p_2 - p_1)^\top \end{bmatrix} p \leq \begin{bmatrix} p_2^\top \\ -p_1^\top \end{bmatrix} (p_1 - p_2) \quad (57)$$

And using the $p_\mathcal{L}$ defined in (48), we define

$$D_{seg}(q) = \begin{cases} D_1(p_1) & (p_1 - p_2)^\top (p_\mathcal{L} - p_2) < 0 \\ D_1(p_2) & (p_2 - p_1)^\top (p_\mathcal{L} - p_1) < 0 \\ D_2(p_\mathcal{L}) & \text{otherwise} \end{cases} \quad (58)$$

where $D_{p\varepsilon}$ and $D_{\mathbf{n}\varepsilon}$ are corresponding constraint function introduced in section II-G and II-H with the constraint set and projection written as:

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

$$\Pi_{seg}(D_{seg}(q)) = 0 \quad (60)$$

J. Convex-polygon-ellipse reachability 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 (48), we define

$$D_{cp}(q) = \begin{bmatrix} D_{seg1} \\ D_{seg2} \\ \vdots \end{bmatrix} \quad (61)$$

with the constraint set and projection written as:

$$\zeta_{cp} = \{q \in \mathbb{R}^{nm} : \|D_{cp}(q)\| = 0\} \quad (62)$$

$$\Pi_{cp}(D_{cp}(q)) = 0 \quad (63)$$

K. Secured planning results and limitation

In this section, we apply our ADMM path planning algorithm to an instance of a map exploration problem (Figure 2a), and test the algorithm in both simulations and an experimental testbed. The environment to be explored, with three agents, is a $10m \times 10m$ region and two forbidden regions *Zone 2*, *3* which is shown in Fig.2b. We assume that the robots can meaningfully collect information on the underlying vector field for data locations no further than $3m$. The maximum velocity constraint v_{max} is set to $0.5m/dt$ a time limit of 20.

We utilize the APMAPF solver introduced in [5] to generate a MAPF plan with co-observation schedule on a grid-world application (Fig.2b) and transform the result to a continuous configuration space, under the map exploration tasks. All robots have the task of collecting sensor information on the underlying vector field, with agent 3 having an additional task to visit the *checkpoint* at time 8. Assuming that the sensors on the robots return data with higher accuracy for locations closer to the agent, the robots should ideally perform a boustrophedon pattern (like the unconstraint result in Figure ??). We first set up an co-observation schedules considering two forbidden regions using the APMAPF algorithm, and add reachability constraints ensure a empty intersection between all robots' reachability regions between co-observations and forbidden regions.

As shown in Fig.2b, no task has been assigned to the agent besides the start and goal location. Agent 1 and 2 need to meet at time 8 and 14, whereas agent 2 and 3 need to meet at time 18. This result is also used as the initial trajectory input for the ADMM solver.

Based on the schedule, reachability for agent 1 in-between the scheduled co-observations are constrained. Notice that additional checkpoint (i.e. stationary security camera) need to be incorporated in order to generated an attack-proof solution. Additional solutions to avoid these checkpoints will be discussed in Chapter ??.

The result of the simulation is shown in Fig.7. The reachability regions are shown as black ellipsoids, and we can show that all of them have empty intersections with *Zone 2* and *3*. Notice that no explicit constraints have been enabled between reachability regions and obstacles, because we assume that

robots have the basic obstacle avoidance capability and can not go through any hard obstacles. Therefore, the intersections between obstacles and ellipsoids, i.e. the case here between agent 3 and zone 1, are tolerable. All constraints have been satisfied and, to the best of their capability, agents have spread out across the map to perform the best possible exploration task.

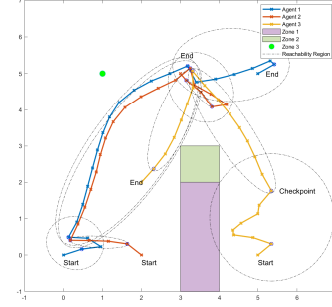


Fig. 7: Simulation result for map exploration task. Reachability regions are shown as black ellipsoids in the result. Zone 1 is obstacle, Zone 2 and Zone 3 are safe zones

1) *Limitations:* Our solution shows that planning with reachability and co-observation is a promising approach to enhance the security of multi-agent systems. However, two main challenges have been identified in [?]. 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 7 is a good example, as it required additional security checkpoint to create secured reachability areas. Secondly, security requirements may come at the cost of overall system performance, as illustrated by the comparison of Figure ?? and 7 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.

III. CROSS-TRAJECTORY CO-OBSERVATION PLANNING

To address the feasibility challenge and performance trade-off of the ADMM problem discussed in Section II-K1, 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 section, we consider only the high-level path planning, while the assignment of the duties in the team will be performed dynamically, 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. Problem overview

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 $q_p(t), p \in \{0, \dots, N_p\}$ represents the reference position of the i -th sub-team at time $t \in \{0, \dots, t_e\}$. 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.

B. Rapidly-exploring Random Trees

In this section, we utilize the RRT* [?] algorithm to address the task of determining path feasibility from a single starting location to multiple destination points (reference trajectories of

other robots in this application). Our objective here is not to optimize specific constraints but to ascertain the existence of feasible paths. While the ADMM based solver II-A3 presented earlier offers a broader range of constraint handling, the RRT* algorithm is better suited to the problem in this section. RRT*'s efficient exploration of the solution space, coupled with its ability to incorporate obstacle constraints, makes it a fitting choice for our specific task.

As an 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). Key functions from RRT* that are called by the algorithm in this section are **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 $(v', v) \in E$

C. 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 $G_q = (V_q, E_q)$.

1) *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 $V_p = \{v_{p1}, \dots, v_{pe}\} \subset V_q$ 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 .

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 $t_1 \in \{t_0, \dots, t_2\}$ 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 $t_3 \in \{t_1, \dots, t_2\}$ 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 8.

2) *Cross-trajectory edges*: After locating checkpoints $\cup_p V_p$ for all sub-teams, we can search for cross-trajectory edges to

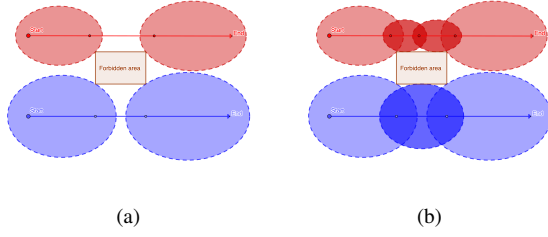


Fig. 8: Checkpoint generation example

connected all checkpoints to trajectories. Cross-trajectory edges represents feasible paths between two reference 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.

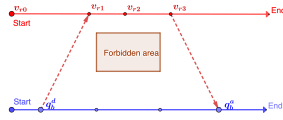


Fig. 9: For checkpoints of the red trajectory, latest departure node q_b^d found for v_{r1} and earliest arrival node q_b^a found for v_{r3} .

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 $V_r = \{(q_{r_i}, t_{r_i})\}$ 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 $t = \text{Cost}(q)/v_{max}$ for a robot to traverse each path. We call all nodes $v_{r_i} \in V_r$ *arrival nodes* 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 nodes* 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 $(v_p \rightarrow v_a)$ and $(v_d \rightarrow v_p)$ are added to E_q . Examples are shown in Figure 9. Full algorithm is shown in Algorithm 1.

Algorithm 1 Cross-trajectory edges generation

procedure CROSSTRAJECTORYEDGES(V)

$V_q \leftarrow V$; $E_q \leftarrow \emptyset$;

for all $v \in V$ **do**

for all q_r that $\mathcal{I}_r \neq \mathcal{I}_v$ **do**

$V_q, E_q \leftarrow \text{ADDEDGES}(q_v, q_r, V_q, E_q)$;

end for

end for

return V_q, E_q

end procedure

procedure ADDEDGES(q_v, q_r, V_q, E_q)

$t_r^d \leftarrow -\infty$; $t_r^a \leftarrow \infty$;

$\mathcal{V}_{goal} = \text{RRT}^*(q_v, q_r)$;

for all $q \in \mathcal{V}_{goal}$ that $\text{Parent}(q) \neq \emptyset$ **do**

if $\text{Cost}(q)/v_{max} < t_q - t_v$ 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 $\text{Cost}(q)/v_{max} < t_v - t_q$ 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**

$v_a \leftarrow (q(t_r^a), t_r^a)$; $v_d \leftarrow (q(t_r^d), t_r^d)$;

$V_q \leftarrow V_q \cup v_a$; $V_q \leftarrow V_q \cup v_d$;

$E_q \leftarrow E_q \cup (v_p \rightarrow v_a)$; $E_q \leftarrow E_q \cup (v_d \rightarrow v_p)$

end if

return V_q, E_q

end procedure

3) *In-trajectory edges*: All checkpoints V_q found in Section III-C1 and III-C2, can be divided by the corresponding sub-team and sorted in ascending order of time into N_p subsets $V_p = \{v_0^p, v_1^p, \dots, v_T^p\}$ where $p \in \{1, \dots, N_p\}$. Consecutive edges in the same subset are added to E_q as in-trajectory edges $(v_i^p \rightarrow v_{i+1}^p)$. Example are shown in Figure 10.

D. 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 robots* in each sub-team

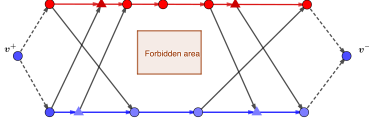


Fig. 10: Example of a two trajectory security graph. Round vertices are checkpoint generated through the heuristic search and triangle vertices are added with the cross-trajectory edges. Additional virtual source v^+ and sink v^- is used later in planning problem.

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 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 $f_{ij}^k \in \{0, 1\}$ 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 $F = [\mathbf{f}_1, \dots, \mathbf{f}_K]$ 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:

$$w_{i,j} = \begin{cases} -w_t & \mathcal{I}_{v_i} = \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ w_c & \mathcal{I}_{v_i} \neq \mathcal{I}_{v_j}, (v_i, v_j) \in E_q \\ 0 & (v_i, v_j) \in E/E_q \end{cases} \quad (64)$$

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) .

$$\min \sum_k \sum_{(+i) \in E} f_{+i}^k - \rho \sum_k \sum_{(ij) \in E} w_{ij} f_{ij}^k \quad (65a)$$

$$s.t. \quad \sum_{\{h:(hi) \in E\}} f_{hi}^k = \sum_{\{j:(ij) \in E\}} f_{ij}^k, \forall k, \forall v \in V_q / \{v^+, v^-\} \quad (65b)$$

$$\sum_k \sum_{\{i:(ij) \in E\}} f_{ij}^k \geq 1, \forall v_j \in \{V_p^s\} \quad (65c)$$

$$f_{ij}^k \in \{0, 1\}, \forall (ij) \in E \quad (65d)$$

The first term in 65a 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 (65b) 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 constraint (65c) 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 polynomial time [?].

E. Co-observation performance

Notice that problem (65) is guaranteed to have a solution for $K = N_p$ where all 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 $K \geq N_p$, it is possible to have a subset of flows $F_E \in F$ that is empty, i.e. $f_{ij} = 0, \forall (v_i, v_j) \in E, f \in F_E$, 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 65a 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 $K = 1$ 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 K does not grow unbounded, penalty parameter ρ need to ensure that the second term for a single flow in (65a) $\rho \sum_{(ij) \in E} w_{ij} f_{ij}^k$ is always smaller than 1.

F. Result and simulation

In this section, we test the proposed method starting from the results in Figure ??, where three trajectories are provided

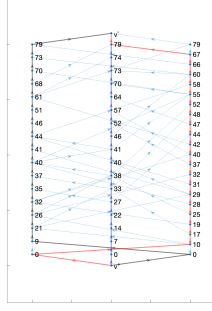


Fig. 11: Security Graph and result for 3 flows

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 $0.5m/dt$.

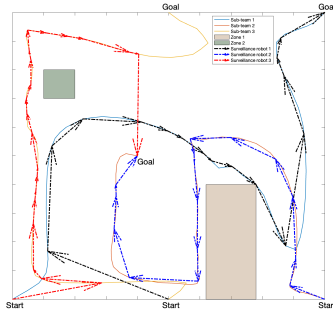


Fig. 12: Result of 3 surveillance agents' plan in workspace

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 the graph G shown in Figure 11, 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 (65) are highlighted in the graph. The planning result in the workspace is shown in 12 as dash-dotted arrows with the same color used for each flow in 11. 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

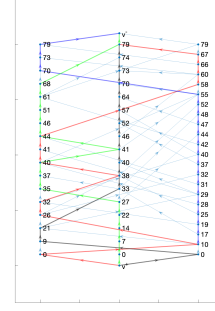


Fig. 13: Security Graph and result for 4 flows.

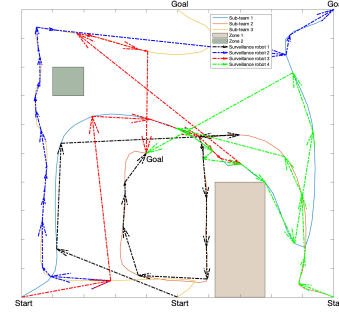


Fig. 14: Result of 3 surveillance agents' plan in work space.

13-14. 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.

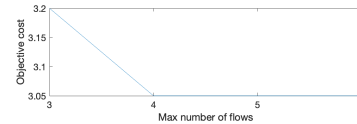


Fig. 15: Plan reach optimality when $\mathcal{K} = 4$, further increase of \mathcal{K} does not increase the cost of objectives.

IV. SUMMARY

In this paper, 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

For subclaim 1)a:

$$H^T H = H^2 = 4uu^T uu^T - 4uu^T + I^2 = I, \quad (66)$$

since $u^T u = 1$.

For subclaim 1)b, let $U = [u \ u_1^\perp \ u_2^\perp]$, where u_1^\perp, u_2^\perp are two orthonormal vectors such that $I = UU^T = uu^T + u_1^\perp (u_1^\perp)^T + u_2^\perp (u_2^\perp)^T$; then, substituting I in (4), we have that the eigenvalue decomposition of H is given by

$$H = U \text{diag}(1, -1, -1)U^T. \quad (67)$$

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

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

$$H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) = Hu \| \nu_{\mathcal{F}} + \nu_{\mathcal{E}} \| = u \| \nu_{\mathcal{F}} + \nu_{\mathcal{E}} \| = \nu_{\mathcal{F}} + \nu_{\mathcal{E}}, \quad (68)$$

and that their difference is flipped under H :

$$H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) = 2uu^T(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}) - (\nu_{\mathcal{F}} - \nu_{\mathcal{E}})^2 = -(\nu_{\mathcal{F}} - \nu_{\mathcal{E}}). \quad (69)$$

Combining (68) and (69) we obtain

$$H\nu_{\mathcal{F}} = \frac{1}{2}(H(\nu_{\mathcal{F}} + \nu_{\mathcal{E}}) + H(\nu_{\mathcal{F}} - \nu_{\mathcal{E}})) = \nu_{\mathcal{E}} \quad (70)$$

B. Proof of proposition 2

From the definition of H in (4), we have

$$\dot{H} = 2(\dot{u}u^T + u\dot{u}^T) \quad (71)$$

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

$$\begin{aligned} H^T \dot{u} &= (2uu^T - I) \frac{(I - uu^T)}{\|u'\|} \dot{u}' \\ &= \frac{1}{\|u'\|} (2uu^T - I - 2uu^T uu^T + uu^T) \dot{u}' \\ &= -\frac{1}{\|u'\|} (I - uu^T) \dot{u}' = -\dot{u} \end{aligned} \quad (72)$$

Inserting $HH^T = I$ in (71), and using ??, we finally have

$$\begin{aligned} \dot{H} &= 2HH^T(\dot{u}u^T + u\dot{u}^T) = 2H(-\dot{u}u^T + u\dot{u}^T) \\ &= -2H[[u]_{\times} \dot{u}]_{\times} \\ &= -2H \left[[u]_{\times} \frac{(I - uu^T)(I - \nu_{\mathcal{F}}\nu_{\mathcal{F}}^T)}{\|u'\| \|\nu_{\mathcal{F}}\|} \dot{\nu}_{\mathcal{F}} \right]_{\times} \\ &= -2H[M\dot{\nu}_{\mathcal{F}}]_{\times}, \end{aligned} \quad (73)$$

which is equivalent to the claim.

C. Proof of proposition 4

To make the notation more compact, we will use $\partial_q f$ instead of $\partial_{[q_1 \atop q_2]} f$ for the remainder of the proof. The differential of (39) can be represented as:

$$\begin{aligned} \dot{\pi}_{p\mathcal{E}} &= \dot{H}^{-1}SH(q_{\text{avoid}} - o) + H^{-1}\dot{S}H(q_{\text{avoid}} - o) \\ &\quad + H^{-1}S\dot{H}(q_{\text{avoid}} - o) + (H^{-1}SH - I)\dot{o} \end{aligned} \quad (74)$$

where

$$\begin{aligned} \dot{S} &= -S^2(Q\dot{s} + s\dot{Q}) \\ &= -S^2(Q\partial_q \dot{s}q - \partial_b Q\partial_q b\dot{q}) \end{aligned} \quad (75)$$

where

$$\partial_b Q = 2\frac{s}{b^3} \text{diag}\{0, 1, 1\} \quad (76)$$

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:

$$\partial_q b = \frac{1}{4b} [q_1 - q_2, q_2 - q_1]^T \quad (77)$$

$$\partial_q o = [I/2, I/2]^T \quad (78)$$

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

$$0 = \dot{F} = 2q^T Q' \dot{q} + q^T \partial_s Q' q \dot{s} + q^T \partial_b Q' q \dot{b} \quad (79)$$

where

$$\partial_s Q' = -\text{diag} \left(\frac{2a^2}{(s+a^2)^3}, \frac{2b^2}{(s+b^2)^3}, \frac{2b^2}{(s+b^2)^3} \right). \quad (80)$$

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

$$\begin{aligned} \dot{s} &= (q^T \partial_s Q' q)^{-1} (2q^T Q' \dot{q} + q^T \partial_b Q' q \dot{b}) \\ &= (q^T \partial_s Q' q)^{-1} (-4q^T Q' H[U\dot{q}]_{\times} (q_{\text{avoid}} - o) \\ &\quad - 2q^T Q' H\dot{o} + q^T \partial_b Q' q \dot{b}) \\ &= (q^T \partial_s Q' q)^{-1} (-4q^T Q' H[q_{\text{avoid}} - o]_{\times} U\dot{q} \\ &\quad - 2q^T Q' H\dot{o} + q^T \partial_b Q' q \dot{b}) \end{aligned} \quad (81)$$

The second term of equation (74) turns into:

$$\begin{aligned} H^{-1}\dot{S}H(q_{\text{avoid}} - o) &= -H^{-1}Q' q \dot{s} - sH^{-1}S^2 \partial_b Q q \dot{b} \\ &= ((q^T \partial_s Q' q)^{-1} H^{-1}Q' q q^T (4Q' H[q_{\text{avoid}} - o]_{\times} U \\ &\quad + 2Q' H\partial_q o - \partial_b Q' q q \partial_q b) - sH^{-1}S^2 \partial_b Q q \partial_q b) \dot{q} \end{aligned} \quad (82)$$

Thus equation (74) could be written as:

$$\begin{aligned} \dot{\pi}_{p\mathcal{E}} &= (-2H[SH(q_{\text{avoid}} - o)]_{\times} U \\ &\quad + ((q^T \partial_s Q' q)^{-1} H^{-1}Q' q q^T (4Q' H[q_{\text{avoid}} - o]_{\times} U \\ &\quad + 2Q' H\partial_q o - \partial_b Q' q q \partial_q b) - sH^{-1}S^2 \partial_b Q q \partial_q b) \\ &\quad - 2H^{-1}SH[q_{\text{avoid}} - o]_{\times} U \\ &\quad + (H^{-1}SH - I)\partial_q o) \dot{q}, \end{aligned} \quad (83)$$

from which the claim follows.

D. Proof of proposition 5

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

$$\dot{d}_{\mathcal{E}} = -n^T \partial_q o \dot{q} \quad (84)$$

$$\begin{aligned} \dot{d}_{\mathcal{E}t} &= (\dot{n}_{\mathcal{E}}^T Q^{-1} n_{\mathcal{E}} + n_{\mathcal{E}}^T \dot{Q}^{-1} n_{\mathcal{E}} + n_{\mathcal{E}}^T Q^{-1} \dot{n}_{\mathcal{E}}) / \sqrt{n_{\mathcal{E}}^T Q^{-1} n_{\mathcal{E}}} \\ &= (\sqrt{n_{\mathcal{E}}^T Q^{-1} n_{\mathcal{E}}})^{-1} (-2n^T H[Q^{-1} n_{\mathcal{E}}]_{\times} U \\ &\quad + n_{\mathcal{E}}^T \partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2n_{\mathcal{E}} Q^{-1} H[n]_{\times} U) \dot{q} \end{aligned} \quad (85)$$

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

$$p_{\mathcal{L}} = \frac{d_{\mathcal{E}} Q^{-1} n_{\mathcal{E}}}{d_{\mathcal{E}t}^2}, \quad (86)$$

we have

$$\begin{aligned} \dot{p}_{\mathcal{L}} &= \left(\left(-\frac{d_{\mathcal{E}t} n^T \partial_q o - 2d_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^3} \right) Q^{-1} n_{\mathcal{E}} \right. \\ &\quad \left. + \frac{d_{\mathcal{E}} \partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2d_{\mathcal{E}} Q^{-1} H[n]_{\times} U}{d_{\mathcal{E}t}^2} \right) \dot{q} \end{aligned} \quad (87)$$

$$\begin{aligned} \dot{p}_1 &= \left(-\frac{Q^{-1} n_{\mathcal{E}} \partial_q d_{\mathcal{E}t}}{d_{\mathcal{E}t}^2} \right. \\ &\quad \left. + \frac{\partial_b Q^{-1} n_{\mathcal{E}} \partial_q b - 2Q^{-1} H[n]_{\times} U}{d_{\mathcal{E}t}} \right) \dot{q} \end{aligned} \quad (88)$$

subtracting \dot{q} from (87) and (88), we can derive the result shown in (48)

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.