Ziqi Yang, Roberto Tron [[1]](#footnote-20) [[2]](#footnote-21) [[3]](#footnote-22)

Abstract

We address security challenges in multi-robot systems (MRS) where adversaries may take control of compromised robots to gain unauthorized physical access to forbidden regions. We propose a novel multi-robot optimal planning algorithm that integrates mutual observations and introduces *reachability constraints* and *co-observations* between robots to enhance security. Our formulation can guarantee that, even with adversarial movements, compromised robots cannot breach forbidden regions without missing scheduled co-observations. We use ellipsoidal over-approximations to reachability regions for efficient intersection checking and gradient computation. Furthermore, to enhance system resilience and tackle feasibility challenges, we introduce the notion of *sub-teams*; these cohesive units replace individual robots along each route; we plan *cross-trajectory co-observation* that use redundant robots to switch between routes, securing multiple sub-teams without requiring modifications to the plans. We formulate cross-trajectories plans by solving a *network-flow vertex path coverage problem* on the *checkpoint graph* generated from the original unsecured MRS trajectories, providing the same security guarantees against plan-deviation attacks. We demonstrate the effectiveness and robustness of our proposed algorithm through simulations, which significantly strengthen the security of multi-robot systems in the face of adversarial threats.

Multi-robot system, cyber-physical attack, trajectory optimization, reachability analysis, network flow

# Introduction

Multi-robot systems (MRS) have found wide applications in various fields. While offering numerous advantages, the distributed nature and dependence on network communication render the MRS vulnerable to cyber threats, such as unauthorized access, malicious attacks, and data manipulation . This paper addresses a specific scenario in which robots are compromised by attacker and directed to *forbidden regions*, which may contain security-sensitive equipment or human workers. Such countering deliberate deviations, termed *plan-deviation attacks*, are identified and addressed in previous studies . As a security measure, we utilized onboard sensing capabilities of the robots to perform inter-robot co-observations and check for unusual behavior. These mutual observation establish a co-observation schedule alongside the path, ensuring that any attempts by a compromised robot to violate safety constraints (such as transgressing forbidden regions) would break the observation plan and be promptly detected.

A preliminary version of the paper was presented in . Extending the grid-world solution from to continuous configuration spaces, we incorporate the co-observation planning as constraints in the alternating direction method of multipliers (ADMM)-based trajectory optimization solver to accomodate for more flexible objectives. In this paper, we incorporate additional reachability analysis during the planning phase, incorporating constraints based on sets of locations that agents could potentially reach, referred to as *reachability regions*, as a novel perspective to the existing literature. This approach enforces an empty intersection between forbidden regions and the reachability region during trajectory optimization, preventing undetected attacks if an adversary gains control of the robots. We utilize an ellipsoidal boundary to constrain the search space and formulate the ellipsoid as the reachability region constraint. We present a mathematical formulation of reachability regions compatible with the solver in as a spatio-temporal constraint.

However, such plans are not guaranteed to exist, and the secured trajectory always come at the cost of overall system performance. As an extension of the previous works , we also address the feasibility and optimality challenges. For a MRS with unsecured optimal trajectories (without security constraints), we introduce redundant robots and formed them into *sub-teams* with the original ones. These redundant robots are assigned the task of establishing additional co-observations, termed *cross-trajectory co-observations*, within and across different sub-teams. The proposed algorithm focuses on the movement plans for the additional robots, distinct from those dedicated to task objectives, indicating when they should stay with their current sub-team and when they should deviate to join another. This strategy allows sub-teams to preserve the optimal unsecured trajectories (as illustrated in [1](#fig:cross-traj-comparison-set)) without requiring the entire *sub-team* to maintain close proximity to other teams during co-observation events. The cross-trajectory co-observation problem is transformed as a multi-agent path finding problem on roadmap (represented as directed graph) and solved as a network flow problem .



[[fig:example-plan-dev]](#fig:example-plan-dev) Attacker deviate the red robot into forbidden region.[[fig:example-co-observation]](#fig:example-co-observation) Red and blue robot tasked to co-observe each other during task resulting a new secured trajectory (solid lines) replacing the optimal ones (dashed line). [[fig:example-cross-traj]](#fig:example-cross-traj) The blue team sends one robot to observe the red team (solid blue line) while having the rest of the robots following the optimal trajectory.

***Related research***. Trajectory planning problem of MRS systems remained as a subject of intense study for many decades, and optimization is a common approaches in such area. Optimization based approaches are customizable to a variable of constraints (e.g. speed limit, avoid obstacles) and task specifications (e.g. maximum surveillance coverage, minimal energy cost). In contrast to our work in [2](#sec:ADMM-planning), many motion planning tasks require a non-convex constraint problem formulation, most contributors focused on convex problems, and only allow for a few types of pre-specified non-convex constraints through convexification. Several optimization techniques like mixed integer programs with quadratic terms (MIQP)  and ADMM  have been used to reduce computational complexity, and to incorporate more complex non-convex constraints.

Graph-based search is another extensively explored approach in trajectory planning problems, such as formation control and Multi-Agent Path Finding (MAPF) problems . In traditional MAPF formulations, environments are abstracted as graphs, with nodes representing positions and edges denoting possible transitions between positions and solved as a network flow problem . This formulation allows for the application of combinatorial network flow algorithms and linear program techniques, offering efficient and more flexible solutions to the planning problem.

Reachability analysis is essential for security and safety verification in cyber-physical systems (CPSs) , often involving an over-approximation of reachable space to verify safety properties. Geometric methods like zonotopes and ellipsoids are commonly used to compactly enclose reachability sets . For online safety assessments against cyber attacks, computed reachable CPS states under attacks and compared them with a safe region based on state estimation. By incorporating an additional security measure that provides two secured states, the reachability region in our work is over-approximated using ellipsoids. This is inspired by the ellipsoidal *heuristic sampling domain* in for the algorithm to simplify the sampling region between start and goal locations. Ellipsoids are also used in other path planning methods like Iterative Regional Inflation by Semidefinite programming (IRIS) and the Safe Flight Corridor (SFC) . Differs from reachability region that requires mapping all reachable states given several known states, the ellipsoids of IRIS and SFC focus on approximate the safe collision-free space rather than addressing security applications.

***Paper contributions***. Two main contributions have been presented.

* We present an innovative method to integrate reachability analysis into the ADMM-based optimal trajectory solver for multi-robot systems, preventing attackers from executing undetected attacks by simultaneously entering forbidden regions and adhering to co-observation schedules.
* We introduce additional robots to form *sub-teams* for both intra-sub-team and cross-sub-team co-observations. A new co-observation planning algorithm is formulated that can generate a resilient multi-robot trajectory with a co-observation plan that still preserve the optimal performance against arbitrary tasks. We also find the minimum redundant robots required for the security.

The rest of this paper is organized as follows. [2](#sec:ADMM-planning) introduces the ADMM-based optimal trajectory solver and introduce the security constraints. [3](#sec:cross-trajectory) introduces the enhanced security planning algorithm through cross-trajectory co-observation. [4](#sec:summary) concludes this article.

***Notation***. In this paper, we use non-bold symbols like to denote single-agent states (e.g. ) and scalers, while bold symbols represent aggregated states involving single robot or multiple robots (e.g. ).

# Secured Multi-robot trajectory planning

We formulate the planning problem as an optimal trajectory optimization problem to minimize arbitrary smooth objective functions. We denote the trajectory as where is the waypoint of agent in a dimensional state space and is the time horizon. For a total of robots, trajectories can be represented as an aggregated vector , where denotes the vertical stacking operation. The overall goal is to minimize or maximize an objective function under a set of nonlinear constraints described by a set , which is given by the intersection of spatio-temporal sets given by traditional path planning constraintsand the security constraints (co-observation schedule, reachability analysis). Formally:

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

**Example 1**. *Robots are tasked to navigate an unknown task space, collecting sensory data to reconstruct a field (see*[*[fig:illustration]*](#fig:illustration)*). The space is modeled as grid points, each with an associated value. The goal is to find paths that minimize uncertainty in reconstructing the field. Each grid point is associated with a Kalman Filter (KF) to track uncertainty through its covariance , updated based on measurements taken by robots along the trajectory , where measurement quality, modeled by a Gaussian radial basis function, is higher near the robot. The optimization objective is to minimize the maximum uncertainty (detailed in ).*



(a) Grid world example of a task planned in a grid world. Blue grids are obstacles, orange and green grids are forbidden regions.

To incorporate the security against malicious deviations (through co-observation and reachability analysis) as constraints in the optimal trajectory planning problem, we define the security as:

A multi-robot trajectory plan is secured against plan-deviation attacks if it ensures that any potential deviations to these forbidden regions will cause the corresponding robot to miss their next co-observation with other robots.

## Preliminaries

### Differentials

We define the differential of a map at a point as the unique matrix such that

where is a smooth parametric curve such that with any arbitrary tangent . For brevity we will use for and for .

With a slight abuse of notation, we use the same notation for the differential of a matrix-valued function with scalar arguments . Note that in this case [[equ:dt\_to\_dx]](#equ:dt_to_dx) is still formally correct, although the RHS needs to be interpreted as applying as a linear operator to .

### Alternating Directions Method of Multipliers (ADMM)

The basic idea of the ADMM-based solver introduced in is to separate the constraints from the objective function using a different set of variables , and then solve separately in [[eq:general-problem]](#eq:general-problem). More specifically, we rewrite the constraint using an indicator function , and include it in the objective function (details can be found in ). We allow to replicate an arbitrary function of the main variables (instead of being an exact copy in general ADMM formulation), to transform constraint to to allow for an easier projection step in [[eq:z-update]](#eq:z-update). In summary, we transform [[eq:general-problem]](#eq:general-problem) into

$$\label{eq:ADMMSetConstraint\_modified}
\begin{aligned}
&\max\quad \varPhi(\mathbf{q})+\varTheta(\mathbf{z}) \\
& \begin{array}{r@{\quad}c}
s.t.& D(\mathbf{q})-\mathbf{z}=0
\end{array}
\end{aligned}$$

where is a vertical concatenation of different functions for different constraints. This makes each constraint set independent and thus can be computed separately in later updating steps. is chosen that the new constraint set  becomes simple to compute which is illustrated in later secitons.

The update steps of the algorithm are then derived as :

$$\label{eq:ADMMupdate}
\begin{align}
\mathbf{q}^{k+1}&:=\mathop{\mathrm{\arg\!\min}}\_\mathbf{q}(\varPhi(\mathbf{q}^k) +\frac{\rho}{2}\norm{D(\mathbf{q})-\mathbf{z}^k+\mathbf{u}^k}\_2^2),\label{eq:admm-mod-update-q}\\
\mathbf{z}^{k+1}&:=\Pi\_\mathcal{Z}(D(\mathbf{q}^{k+1})+\mathbf{u}^k), \label{eq:z-update} \\
\mathbf{u}^{k+1}&:=\mathbf{u}^k+D(\mathbf{q}^{k+1})-\mathbf{z}^{k+1}, \label{eq:u-update}
\end{align}$$

where is the new projection function to the modified constraint set , represents a scaled dual variable that, intuitively, accumulates the sum of primal residuals

Checking the primal residuals alongside with the dual residuals

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 number of iterations.

In practical application, [[eq:admm-mod-update-q]](#eq:admm-mod-update-q) is solved iteratively via nonlinear optimization solver like *fmincon*, while [[eq:z-update]](#eq:z-update) and [[eq:u-update]](#eq:u-update) have closed form solution.

We now provide the functions , the sets , and the corresponding projection operators for security constraints including co-observation security constraints ([2.2](#sec:co-observation-constraint)), and reachability constraints ([2.5](#sec:ellipsoid-point)-[[eq:region\_ellipsoid\_constraint]](#eq:region_ellipsoid_constraint)). The latter are based on the definition of *ellipse-region-constraint* ([2.3](#sec:reachability)). Formulation of traditional path planning constraints like velocity constraints, convex obstacle constraints can be found in , thus are omitted in this paper.

## Co-observation schedule constraint

The co-observation constraint ensures that two robots come into close proximity at scheduled times to observe each other’s behavior. This constraint is represented as a relative distance requirement between the two robots at a specific time instant, ensuring they are within a defined radius to inspect each other or exchange data.

**ADMM constraint 1** (Co-observation constraint). *$$\begin{aligned}
D(\mathbf{q}) &= \overrightarrow{q\_{aj}q\_{bj}}, \label{eq:coobservation\_constraint}\\
\mathcal{Z}&= \{\mathbf{z}\mid \norm{\mathbf{z}} \leq d\_{max} \},\\
\Pi\_\mathcal{Z}(z) & = \begin{cases}
d\_{max}\frac{\mathbf{z}}{\norm{\mathbf{z}}} &\text{if } \norm{\mathbf{z}} > d\_{max},\\
\mathbf{z}& otherwise,
\end{cases}
\end{aligned}$$ where are the indices of the pair of agents required for a mutual inspection, is the maximum distance between a pair of robots that the secure can be ensured through co-observation.*

The locations and where the co-observation is performed are computed as part of the optimization.

## Definition of ellipsoidal reachability regions

In this section, we define *ellipsoidal reachability regions* with respect to a pair of locations along a trajectory. We then use the ellipsoids to introduce various reachability constraints with respect to other geometric entities, such as points, lines, line segments, and polygons. These constraints, combined with the co-observation constraint [1](#constraint:coobservation), form the basis of the secured ADMM planning presented in [2.1.2](#chapter:ADMM review).

Consider a robot starting from at time and reaching at time . The *reachability region* between and is defined as the set of points in the free configuration space for which there exists a kinematically feasible trajectory containing .

For simplicity, in this paper we consider only a maximum velocity constraint ; the reachability region for can then be over-approximated by the following (the over-approximation is obtained by neglecting physical obstacles):

The *reachability ellipsoid*  is defined as the region , where .

This region is an ellipsoid because it represents the set of points whose sum of distances two *foci* and is less then , where a is the major radius of the ellipsoid.



([[fig:EllipseConstraintExample]](#fig:EllipseConstraintExample)) Black line is the planned trajectory, and are two co-observed locations the robot are expected at given time and , dashed lines show possible trajectory of a compromised robot during unobserved period. Axis of global frame and canonical frame are shown as and respectively. ([[fig:Ellipse-to-point]](#fig:Ellipse-to-point)) For point-ellipsoid constraint, is projected to the areas outside the ellipsoid to . ([[fig:Ellipse-to-plane]](#fig:Ellipse-to-plane)) For plane-ellipsoid constraint, the projection is simplified to the point-ellipse constraint that projecting point outside the ellipse to . ([[fig:Ellipse-to-safezone]](#fig:Ellipse-to-safezone)) For convec-polygon-ellipsoid constraint, the projection either a plane-ellipse constraint (for red region) or a point-ellipse constraint (for brown region).

The condition that a reachability ellipsoid does not intersect with any forbidden region is sufficient to secure the trajectory of the robot between and according to [[def:secured-plan]](#def:secured-plan): any deviation to a point outside will cause the robot to miss a potentia observation at , . [[def:secured-plan]](#def:secured-plan) can then be restated as follows:

A multi-robot trajectory is secured against plan-deviation attacks if there exist a co-observation plan such that the reachability region between each consecutive co-observation does not intersect with any forbidden regions.

## Transformation to canonical coordinates

To use the definition of reachability ellipsoid from the section above in a computational constraints for ADMM, we first apply a differentiable rigid body transformation to reposition the ellipse from the global frame to a canonical frame , where the origin of is at the ellipsoid’s center , and the first axis of is aligned with the foci (see [[fig:EllipseConstraintExample]](#fig:EllipseConstraintExample) for an illustration). From this definition, a unit vector along the first axis of the ellipsoid in the frames and is given by $\nu\_\mathcal{F}= \frac{q\_2-q\_1}{\norm{q\_2-q\_1}}$ and , respectively. The full transformation from coordinates to , and its inverse, are then parametrized by a rotation and a translation as (we drop the subscript and superscript from and to simplify the notation):

where the rotation matrix is a *Householder rotation* (a differentiable linear transformation describing the minimal rotation between the two vectors, see [4.2](#sec:householder) for details). To simplify the notation, in the following, we will consider to be a function of directly, i.e. . More details on [[eq:transformations]](#eq:transformations) are given in [4.1](#apx:transformation).

Reachability constraints are formulated with respect to different types of forbidden regions a point, a plane, a segment, and a convex polygon.

## Point-ellipsoid reachability constraint

For a forbidden region in the shape of a single point ([[fig:Ellipse-to-point]](#fig:Ellipse-to-point)), the constraint is written as:

**ADMM constraint 2** (Point-ellipsoid reachability constraint). *$$\begin{aligned}
D(\mathbf{q}) &= \begin{cases}
\pi\_{p\mathcal{E}}(\mathbf{q}) - q\_{avoid} & q\_{avoid} \in \mathcal{E}, \\
{0} & \textrm{otherwise}.
\end{cases} \label{eq:point\_ellipsoid\_constraint}\\
\mathcal{Z}&= \{q\in\mathbb{R}^{nm}:\norm{D\_p(\mathbf{q})} = 0 \},\\
\Pi\_\mathcal{Z}(\mathbf{z}) & = 0,
\end{aligned}$$*

where is a projection function that returns a projected point of outside the ellipse, i.e., as the solution to

$$\label{eq:ellipse-point-projection}
\pi\_{p\mathcal{E}}=\mathop{\mathrm{\arg\!\min}}\_{q\_p\in\mathcal{E}^c} \quad \norm{q\_{avoid}-q\_p}^2$$

where is the set complement of region .

For cases where , . And for cases where , needs to be projected to the boundary of the ellipse. In the canonical frame , the projected point can be written as:

where can be solved as the root of the level set [[equ:standard-ellipse]](#equ:standard-ellipse):

where

Detailed methods for computing can be found in .

The point-to-ellipse projection function in is then:

In our derivations, we consider only the 3-D case (); for the 2-D case, let : then .

The differential of the projection operator with respect to the foci is given by the following (using as a shorthand notation for )

*Proof.* See [4.3](#proof:Ellipse2PointDiff). ◻

The differential of is the same as the one for .

## Plane-ellipsoid reachability constraint

For a hyperplane shaped forbidden region ([[fig:Ellipse-to-plane]](#fig:Ellipse-to-plane)), the reachability constraint is . When transformed into the canonical frame, the hyperplane can be written as , with .

For every , there exist two planes that are both parallel to and tangential to the ellipse (i.e. resulting in a unique intersection point [[fig:Ellipse-to-plane]](#fig:Ellipse-to-plane)), and . The intersection point can be written as:

where ; intuitively, can be thought as a distance between the tangent plane (or ) and the origin (i.e., the center of the ellipse ). The concept of *tangent interpolation point*s is introduced to characterize the relationship between the plane and the ellipsoid.

The *tangent interpolation point* between the plane and the ellipsoid is defined on the plane by

Intuitively, the point is the point on which is closest to either or . Note that when or , or , respectively. When , the plane and the ellipsoid have at least one intersection, thus violating our desired reachability constraint.

With these definitions, the constraint can be written as:

**ADMM constraint 3** (Plane-ellipsoid reachability constraint). *$$\begin{aligned}
D(\mathbf{q}) &= H^{-1}(q\_{1},q\_2)\pi^\mathcal{E}\_{\mathbf{n}\_\mathcal{E}}(\mathbf{q})+o, \label{eq:plane\_ellipsoid\_constraint}\\
\mathcal{Z}&= \{\mathbf{q}\in\mathbb{R}^{nm}:\norm{D\_{\mathbf{n}}(\mathbf{q})} = 0 \},\\
\Pi\_\mathcal{Z}(\mathbf{z}) & = \overrightarrow{0},
\end{aligned}$$*

where is the projection operator defined as:

The differential of the projection function with respect to the foci and is given by:

where

*Proof.* See [4.4](#proof:dpi_ne_dt). ◻

The differential of ([[eq:plane\_ellipsoid\_constraint]](#eq:plane_ellipsoid_constraint)) can be written as:

## Line-segment-ellipse reachability constraint

As an intermediate step to consider polygon shaped forbidden regions, the reachability constraint for hyperplane segments is studied. For hyperplane with endpoints and ; this segment is defined as:

When the *tangent interpolation point* stays within the segment (i.e. red region in [[fig:Ellipse-to-safezone]](#fig:Ellipse-to-safezone)), the constraint is a plane-ellipse constraint in [2.6](#sec:ellipsoide-plane). Otherwise (i.e. brown region in [[fig:Ellipse-to-safezone]](#fig:Ellipse-to-safezone)), the constraint is a point-ellipse constraint in [2.5](#sec:ellipsoid-point).

**ADMM constraint 4** (Line-segment-ellipsoid reachability constraint). *$$\begin{aligned}
D(\mathbf{q}) &= \begin{cases}
D\_{p\_{1}}(\mathbf{q}) & ({p\_1^{\mathcal{E}}}-{p\_2^{\mathcal{E}}})^\mathrm{T}({p\_{\mathcal{L}}^{\mathcal{E}}}-{p\_2^{\mathcal{E}}})<0\\
D\_{p\_{2}}(\mathbf{q}) & ({p\_2^{\mathcal{E}}}-{p\_1^{\mathcal{E}}})^\mathrm{T}({p\_{\mathcal{L}}^{\mathcal{E}}} -{p\_1^{\mathcal{E}}})<0\\
D\_{p\_{\mathcal{L}}^{\mathcal{E}}}(\mathbf{q}) & \textrm{otherwise}
\end{cases} \label{eq:segment\_ellipsoid\_constraint}\\
\mathcal{Z}&= \{\mathbf{q}\in\mathbb{R}^{nm}:\norm{D(\mathbf{q})} = 0 \},\\
\Pi\_\mathcal{Z}(\mathbf{z}) & = 0,
\end{aligned}$$*

where and are the point-ellipsoid constraint projection function [[eq:point\_ellipsoid\_constraint]](#eq:point_ellipsoid_constraint) with respect to and , and is the plane-ellipsoid constraint [[eq:plane\_ellipsoid\_constraint]](#eq:plane_ellipsoid_constraint). with respect to frame .

## Convex-polygon-ellipse reachability constraint

For reachability constraints with respect to a convex polygon, the first step is to keep all segments that defines the hyperplane outside the reachability ellipsoid. Similar to [[equ:ProjectPoint]](#equ:ProjectPoint), we define

**ADMM constraint 5** (Convex-polygon-ellipsoid reachability constraint). *$$\begin{aligned}
D(\mathbf{q}) &= \begin{bmatrix}D\_{seg1}(\mathbf{q})\\D\_{seg2}(\mathbf{q})\\ \vdots\end{bmatrix} \label{eq:region\_ellipsoid\_constraint}\\
\mathcal{Z}&= \{\mathbf{q}\in\mathbb{R}^{nm}:\norm{D(\mathbf{q})} = 0 \},\\
\Pi\_\mathcal{Z}(\mathbf{z}) & = 0,
\end{aligned}$$*

where are the constraint functions for all line segments used to define the convex polygon region. This constraint needs to be supplemented with a convex obstacle constraint for the polygon (i.e. keep foci waypoints outside the region, introduced in ) to prevent cases where the ellipse is a subset of the region.

## Secured planning results and limitation

We test the newly introduced security constraints with [1](#example:map_exploration), illustrated in [3](#fig:example-application) in both simulations and on an experimental testbed, involving a 3-robot system assigned to explore a region with one obstacle, *Zone 1*, and two forbidden zones, *Zone 2* and *Zone 3*. The maximum velocity constraint , time horizon .

We employ the attack-proof MAPF(APMAPF) solver on a 8 by 8 grid world with similar setup to generate a MAPF plan with a co-observation schedule in a grid-world application (Fig.[[fig:Grid-example-application]](#fig:Grid-example-application)). This result is transformed to a continuous configuration space and serves as the initial trajectory input for the ADMM solver with additional task function targeting the map exploration task. Co-observation schedules are set up using the APMAPF algorithm for two forbidden regions. Reachability constraints are added to ensure an empty intersection between all robots’ reachability regions during co-observations and the forbidden regions. It is important to emphasize that an APMAPF solution does not guarantee existence, nor does it ensure a successful transition to the continuous configuration space. In this cases, while an APMAPF solution may be found, the secured, attack-proof solution becomes infeasible in the continuous setting because the robots’ mobility is no longer restricted to adjacent grids. Additional security measures, such as stationary security cameras or surveillance robots, need to be incorporated. For instance, in this scenario, we deploy a security camera as additional checkpoint to observe agent 3 at time to ensure security.

The simulation result, shown in Fig. [[fig:ReachabilitySimulation]](#fig:ReachabilitySimulation), displays reachability regions as black ellipsoids, demonstrating empty intersections with Zones 2 and 3. Explicit constraints between reachability regions and obstacles are not activated, assuming basic obstacle avoidance capabilities in robots. The intersections between obstacles and ellipsoids, as observed between agent 3 and Zone 1, are deemed tolerable. All constraints are satisfied, and agents have effectively spread across the map for optimal exploration tasks.



Trajectory design of a map exploration task forthree-robot system based on result in grid world result [[fig:Grid-example-application]](#fig:Grid-example-application). Zone 1 is obstacle, Zone 2 and Zone 3 are safe zones. [[fig:SecurityBreak]](#fig:SecurityBreak) shows the unsecured trajectory design in continuous world optimized with respect to a map exploration task where Zone 1 and 2 are considered together as one obstacle. A potential path that could break the security by entering region 1 is highlighted in dashline. [[fig:ReachabilitySimulation]](#fig:ReachabilitySimulation) shows the secured trajectory with the incorporation of co-observation and reachability constraints.

### Limitations

Our solution demonstrates the potential of planning with reachability and co-observation to enhance the security of multi-agent systems.However, two primary challenges need to be addressed. Firstly, achieving a co-observation and reachability-secured plan is not always feasible, particularly when obstacles or restricted regions separate the robots, preventing them from establishing co-observation schedules or finding reachability-secured paths. For example, agent 3 in [[fig:ReachabilitySimulation]](#fig:ReachabilitySimulation) requires additional security measures to create secured reachability areas. Secondly, security requirements can impact overall system performance, as illustrated by the comparison between [[fig:SecurityBreak]](#fig:SecurityBreak) and [[fig:ReachabilitySimulation]](#fig:ReachabilitySimulation). The introduction of security constraints resulted in the top left corner remaining unexplored by any robots. This trade-off between security and system performance is particularly significant as system performance is a key factor in the decision of multi-agent system deployments. These challenges are further addressed in [3](#sec:cross-trajectory), ensuring the effective integration of reachability and co-observation in securing multi-agent systems.

# Cross-trajectory co-observation planning

To address the feasibility and performance trade-offs, we propose to form *sub-teams* on each route, and setup additional co-observations both within the sub-team and across different sub-teams. These *cross-trajectory co-observations* allow sub-teams to obtain trajectories that are closer to the optimal (as shown in [1](#fig:cross-traj-comparison-set)), because they do not require the entire *sub-team* to meet with other teams for co-observations, thus providing more flexibility.

## Problem overview

We start with an unsecured MRS trajectory with routes (the ADMM-based planner in [2](#sec:ADMM-planning) without security constraints is used, but other planners are applicable). Here, the state represents the reference position of the -th sub-team at time . Introducing redundant robots into the MRS, we assume a total of robots are available and organized into *sub-teams* through a time-varying partition where robots in each *sub-team* share the same nominal trajectory. To ensure the fulfillment of essential tasks, at least one robot is assigned to adhere to the reference trajectory. The remaining redundant robots are strategically utilized to enhance security. These robots focus on co-observations either within their respective sub-teams, or when necessary, deviate and join another sub-team to provide necessary co-observations. The objective of this problem is to formulate a strategy for this new co-observation strategy, named *cross-trajectory co-observation* that the resulting MRS plan meet the security in [[rmk:revised-security]](#rmk:revised-security) while minimizing the number of additional robots required.

Our strategy is based on the following concept:

The -th *checkpoint* for sub-team is defined as a location and time . It representes either the first or last instance where a robot is co-observed by a teammates, and serves as a point where the robot can leave or join a new team.

For simplicity, let denote the sub-team to which belongs. To ensure the security of the reference trajectory, the reachability region between consecutive checkpoints must avoid intersections with forbidden regions. This requirement can be formally stated as:

A set of checkpoints (arranged in ascending order of ) can secure the reference trajectory for sub-team , if for every , where is the union of all forbidden regions.

The planned trajectory is represented as a directed *checkpoint graph* , where is the set of verticies representing waypoints on the planned trajectory and is the set of edges representing feasible paths connecting . The components of the checkpoint graph are defined as follows:

Checkpoints

where additional robots are required for security through co-observation.

Tranectory edges

representing the original planned path of the primary robot.

Connection vertices

representing non-checkpoint locations linked via .

Cross-trajectory edges

representing feasible paths where robots deviate from the original trajectory to join other teams or provide co-observation support, at least one endpoint be a checkpoint.

We assume that the planned trajectory for each sub-team has a robot operating on it. The co-observation planning problem is then transformed into a network multi-flow problem by modelling the additional robots as flows traveling through . These flows either follow the planned trajectory via trajectory edges or switch teams via cross-trajectory edges to ensure valid and secured paths. Additionally, they are required to visit every checkpoint to meet the security requirements introduced in [[rmk:checkpoints]](#rmk:checkpoints). This formulation reduces the co-observation problem to finding valid flow patterns in . The problem is then formulated into linear objectives and constraints, enabling the use of Mixed-Integer Linear Programming (MILP) techniques to efficiently compute feasible solutions.

This section presents the two components of our approach: constructing the checkpoint graph based on unsecured multi-robot trajectories, and the formulation and solution of the network multi-flow problem.

## Rapidly-exploring Random Trees

To find edges between different nominal trajectories, we use the   algorithm to find paths from a waypoint on one trajectory to multiple destination points (i.e., reference trajectories of other sub-teams, in our method). As a optimal path planning algorithm, returns the shortest paths between an initial location and points in the free configuration space, organized as a tree. We assume that the generated paths can be travelled in both directions (this is used later in our analysis). Key function from is used during constructions of the checkpoint graph, specifically

This function returns a total travel distance to the unique path from the initial position to .

Our objective here is to ascertain the existence of feasible paths instead of optimize specific tasks. While the ADMM based solver [2](#sec:ADMM-planning) presented earlier offers a broader range of constraint handling, RRT\*’s efficient exploration of the solution space, coupled with its ability to incorporate obstacle constraints, makes it a fitting choice for building the checkpoint graph.

## Checkpoint graph construction

In this section, we define and search for security checkpoints and how to use to construct the checkpoint graph . A heuristic approach is provided 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 , to , with , as *start* and *end verticies*. Then, we search for the largest such that the reachability ellipsoid between and has no overlap with any of the forbidden areas, i.e. . Then is added to . Simultaneously, we search backward to find with the smallest that meets the reachability requirement . Then is added to . Finally, we set , and repeat the same process. The search is stopped when or . A toy example is shown in [4](#fig:checkpoint-generate).



### Cross-trajectory edges

To enable co-observation cross different sub-team at checkpoints, we search for available connection paths (*cross-trajectory edges*) between checkpoints on different trajectories, allowing robots deviate from one sub-team to perform co-observation with a different sub-team.

Cross-trajectory edges define viable paths between two reference trajectories, where and at least one of and correspond to a security checkpoint . The cross-trajectory edges must also adhere to reachability constraints to ensure that no deviations into forbidden regions can occur while switching trajectories.



([[fig:two-edges]](#fig:two-edges)) Latest departure node found for and earliest arrival node found for . ([[fig:security-graph-generate]](#fig:security-graph-generate)) A full checkpoint graph, where circles are checkpoints , triangles are connection nodes , colored arrows (red and blue) are trajectory edges and black arrows are cross-trajectory edges .

We first find trees of feasible paths between each security checkpoint and all other trajectories using position information alone (ignoring, for the moment, any timing constraint). More precisely, for each checkpoint , we use   to find all feasible, quasi-optimal paths from to all the waypoints on the reference trajectories of all other sub-teams . Then, to prune these trees, we consider the time needed to physically travel from one trajectory to the other while meeting other robots at the two endpoints. This is done by calculating the minimal travel time for a robot to traverse each path. Two types of connecting nodes can be found.

Arrival nodes

are waypoints on where robots from sub-team , deviating at , can meet with sub-team at . For these, must have found a path from to with , and .

For each trajectory , we define the *earliest arrival node* from to as the arrival node characterized by the minimum discovered.

Departure nodes

are the waypoints on such that a robot from sub-team , deviating from , can meet with robots in sub-team at . For these nodes, must find a path from to for if and .

For each trajectory , we define the *latest departure node* as the departure node characterized by the maximum discovered.

The set contains all the *latest departure* and *earliest arrival* nodes (if they are not identified as checkpoints already); the corresponding paths are added as cross-trajectory edges . A toy example is shown in [[fig:two-edges]](#fig:two-edges).

### In-trajectory edges

We group by sub-teams and arrange them in ascending order of their timestamps . For each sub-team, consecutive vertices and are connected by adding *in-trajectory edges* to , representing corresponding segment of the planned trajectory. Examples are shown in [[fig:security-graph-generate]](#fig:security-graph-generate).

## Co-observation planning problem

In this section, we formulate the cross-trajectory planning problem as a network multi-flow problem, and solve it using mixed-integer linear programs (MILP). We assume that robots are dedicated (one in each sub-team) to follow the reference trajectory (named *reference robots*). The goal is to plan the routes of the additional *cross-trajectory robots* dedicated to cross-trajectory co-observations, and potentially minimize the number of cross-trajectory robots needed.

Note that we assign fixed roles to robots for convenience in explaining the multi-flow formulation. In practice, after a cross-trajectory robot joins a team, it is considered interchangable, and could switch roles with the reference robot of that trajectory.

To formulate the problem as a network multi-flow problem, we augment the checkpoint graph to a flow graph. A *virtual source* node and a *virtual source* node is added to the vertices . Additional directed edges from to all the start vertices, and from all end vertices to are added with and representing the start and end vertices of sub-team (dashed arrows in [[fig:example-cross-traj]](#fig:example-cross-traj)).

The path of a robot all starts from and ends at , and are represented as a flow vector , where is an indicator variable representing whether robot ’s path contains the edge . The planning problem can be formulated as a vertex path cover problem on , i.e., as finding a set of paths for cross-trajectory robots such that every checkpoint in is included in at least one path in (to ensure co-observation at every checkpoint as required by [[rmk:revised-security]](#rmk:revised-security)).

Technically, we can always create a trivial schedule that involves only co-observations between members of the same team; this, however, would make the solution more vulnerable in the case where multiple agents are compromised in the same team. While in this paper we explicitly consider only the single attacker scenario, multi-attacks can be potentially handled by taking advantage of the *decentralized blocklist protocol* introduced in . For this reason, we setup the methods presented below to always prefer *cross-trajectory co-observation* when feasible.

Finally, edges from the virtual source and to the virtual sink should have zero cost, to allow robots to automatically get assigned to the starting point that is most convenient for the overall solution (lower cost when taking cross-trajectory edges). These requirements are achieved with the weights for edges defined as:

where .

With the formulation, the planning problem is written as an optimization problem, where the optimization cost balances between the co-observation performance and total number of flows (cross-trajectory robots) needed:

where, for convenience, we used to represent the edge , and to represent the edge .

The first term in the cost [[eq:flow-cost]](#eq:flow-cost) represents the total number of robot used, and the second term 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 is a manually selected penalty parameter to balance between the two terms. The flow conservation constraint [[eq:FlowBalanceConstraint]](#eq:FlowBalanceConstraint) ensures that the amount of flow entering and leaving a given node is equal (except for and ). The flow coverage constraint [[eq:FlowCoverage]](#eq:FlowCoverage) ensures that all checkpoints have been visited and can support a co-observation. The checkpoint graph is acyclic, placing this problem in complexity class ; thus, it can be solved in polynominal time .

## Co-observation performance

The network flow problem [[eq:flow-coverage-problem]](#eq:flow-coverage-problem) is guaranteed to have a solution for where all robots follows the reference trajectory (). It is possible for the resulting flows to have a subset of flows that is empty, i.e. , ; these flow will not increase the cost and can be discarded from the solution.

The constant selects the trade-off between the number of surveillance robots and security performance. An increase in robots generally enhances security via cross-trajectory co-observations; this also increases the complexity of the coordination across robots. In order to identify the minimum number of robots necessary, we propose an iterative approach in which we start with and gradually increase it until contains an empty flow, indicating the point where further additions of robots do not improve performance.

The value of row is upper bounded such that the second term for each single flow in [[eq:flow-cost]](#eq:flow-cost) is always smaller than one, i.e., . Otherwise, the iteration continues indefinitely as adding additional flow introduce a negative term which always makes the cost [[eq:flow-cost]](#eq:flow-cost) smaller.

## Result and simulation



[[fig:previous\_result\_plan]](#fig:previous_result_plan) showcase an unsecured 4-robot map exploration task. The cross-trajectory co-observation plan is shown as dotted arrows on top of the original plan in [[fig:previous\_result\_plan]](#fig:previous_result_plan). The checkpoint graph and resulting flows are shown in [[fig:previous\_result\_graph]](#fig:previous_result_graph).

We first test the proposed method for the example application in [[fig:example-application,fig:ReachabilitySimulation]](#X3fece3d92f30fefd799c556da67ced88f12bc44), using the same setup in [2.9](#sec:ADMM-simulation). Using the parameters , and , the result returns a total of surveillance robots with cross-trajectory plan shown in [5](#fig:3-team-ctco). The flows derived from the solution of the optimization problem [[eq:flow-coverage-problem]](#eq:flow-coverage-problem) are highlighted in the graph [[fig:previous\_result\_graph]](#fig:previous_result_graph), where each horizontal line represents the original trajectory of a sub-team and the number on each vertex represents the corresponding time . The planning result in the workspace is shown in [[fig:previous\_result\_plan]](#fig:previous_result_plan) as dash-dotted arrows with the same color used for each flow in [[fig:previous\_result\_graph]](#fig:previous_result_graph). Compared with the result in [[fig:ReachabilitySimulation]](#fig:ReachabilitySimulation), it is easy to see that there is a significant improvement in map coverage in [[fig:previous\_result\_plan]](#fig:previous_result_plan). At the same time, unsecured deviations to the forbidden regions of robots are secured through cross-trajectory observations.

The problem shows no solution for . For cases , problem return the optimal result as shown in [6](#fig:Cross-trajectory-result). If we further increase , we do not get a better result; instead, the planner will return four flows with the rest flows empty.

We then test the proposed method for a 4-team and 7-team case with result shown in [6](#fig:Cross-trajectory-result), where four trajectories are provided for a map exploration task with no security related constraints (co-observation schedule and reachability). We have a task space, three forbidden regions (rectangle regions in [[fig:Result-plan]](#fig:Result-plan)) and robots with a max velocity of .



Cross-trajectory co-observation result of a 4 sub-teams task ([[fig:Result-plan]](#fig:Result-plan)) and a 7 sub-teams task [[fig:Result-plan-7-team]](#fig:Result-plan-7-team)

# Summary

This paper introduces security measures for protecting Multi-Robot Systems (MRS) from plan-deviation attacks. We establish a mutual observation schedule to ensure that any deviations into forbidden regions break the schedule, triggering detection. Co-observation requirements and reachability analysis are incorporated into the trajectory planning phase, formulated as constraints for the optimal problem. In scenarios where a secured plan is infeasible or enhanced task performance is necessary, we propose using redundant robots for cross-trajectory co-observations, offering the same security guarantees against plan-deviation attacks. However, the time-sensitive and proximity-dependent nature of co-observation necessitates a focus on collision avoidance during planning. Future work will integrate collision avoidance and explore dynamic duty assignments within sub-teams, increasing the challenge for potential attackers.

## Transformation of a reachability ellipsoid to canonical frame

The ellipse expressed in is given by , with foci in defined as , and semi-axis distance $c=\frac{\norm{q\_2-q\_1}}{2}$.

The *reachability ellipsoid in the canonical frame* is defined as as the zero level set of the quadratic function

where

and . The ellipse parameters , represent the lengths of the major axes.

The original ellipse in can be expressed as the zero level set of the quadratic function

*Proof.* The claim follows by substituting [[eq:transformations]](#eq:transformations) into [[equ:standard-ellipse]](#equ:standard-ellipse), and from the definition of and . ◻

## Householder rotations

We define a differentiable transformation to a canonical ellipse that is used in the derivation of the reachability constraints in [[sec:reachability,sec:ellipse-region-constraint]](#X0090afe133958e3f4a84bfd0b44f3d4ace218dd). This transformation includes a rotation derived from a modified version of Householder transformations . 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 plane.

Let and be two unitary vectors ($\norm{\nu\_1}=\norm{\nu\_2}=1$). Define the normalized vector $u = \frac{\nu\_1+\nu\_2}{\norm{\nu\_1+\nu\_2}}$, the *Householder rotation* is defined as

Here is a rotation mapping to , as shown by the following.

The matrix has the following properties:

1. It is a rotation, i.e.
   1. ;
   2. .
2. .

*Proof.* For subclaim [[it:orthonormality]](#it:orthonormality):

since .

For subclaim [[it:determinant]](#it:determinant), let , where are two orthonormal vectors such that ; then, substituting in [[eq:H definition]](#eq:H definition), we have that the eigenvalue decomposition of is given by

Since the determinant of a matrix is equal to the product of the eigenvalues, .

For subclaim [[it:transformation]](#it:transformation), first note that . It follows that the sum of and is invariant under :

$$\label{equ:H(u1+u2)}
H(\nu\_1+\nu\_2)=H u \norm{\nu\_1+\nu\_2}
= u\norm{\nu\_1+\nu\_2} =\nu\_1+\nu\_2,$$

and that their difference is flipped under :

Combining ([[equ:H(u1+u2)]](#equ:H(u1+u2))) and ([[equ:H(u1-u2)]](#equ:H(u1-u2))) we obtain

 ◻

We compute the differential of implicitly using its definition [[equ:dt\_to\_dx]](#equ:dt_to_dx). We use the notation to denote the matrix representation of the cross product with the vector , i.e.,

such that for any .

Let represent a parametric curve. Then

where the matrix is given by

$$M=[u]\_{\times} \frac{ \left( I - u u^\mathrm{T}\right) \left( I- \nu\_1 \nu\_1^\mathrm{T}\right)} {\norm{u'} \norm{\nu\_1}}.$$

*Proof.* From the definition of in [[eq:H definition]](#eq:H definition), we have

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

$$\begin{gathered}
H^\mathrm{T}\dot u = (2u u^\mathrm{T}-I)\frac{ \left( I - u u^\mathrm{T}\right) } {\left\|u'\right\|} \dot u' \\
=\frac{1}{\norm{u'}} (2u u^\mathrm{T}-I-2u u^\mathrm{T}u u^\mathrm{T}+u u^\mathrm{T})\dot{u}'\\
= -\frac{1}{\norm{u'}} (I-u u^\mathrm{T})\dot u'
= -\dot u
\end{gathered}$$

Inserting in ([[equ:H\_dot original]](#equ:H_dot original)), we have

which is equivalent to the claim. ◻

## Proof of proposition [[prop:Ellipse2PointDiff]](#prop:Ellipse2PointDiff)

To make the notation more compact, we will use instead of for the remainder of the proof. The differential of [[equ:Point2EllipseProjection]](#equ:Point2EllipseProjection) can be represented as:

where

where

To compute the derivative , we need the expression of , and ; the first two can be easily derived using the equations above:

In order to get , we use the fact that for all ; hence , and . We then have:

where

By moving term to the left-hand side we can obtain:

The second term of equation ([[equ:dPi\_dt]](#equ:dPi_dt)) turns into:

Thus equation [[equ:dPi\_dt]](#equ:dPi_dt) could be written as:

from which the claim follows.

## Proof of proposition [[prop:dpi\_ne\_dt]](#prop:dpi_ne_dt)

We first need to derive and

Next, we need to derive , and . Since could be written as

we have

subtracting from ([[equ:dpl\_dt]](#equ:dpl_dt)) and ([[equ:dplt\_dt]](#equ:dplt_dt)), we can derive the result shown in ([[equ:ProjectPoint]](#equ:ProjectPoint))

1. 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). [↑](#footnote-ref-20)
2. Ziqi Yang is with the Department of Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: zy259@bu.edu). [↑](#footnote-ref-21)
3. Roberto Tron is with the Faculty of Mechanical Engineering and Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: tron@bu.edu). [↑](#footnote-ref-22)