2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Multi-Agent Path Planning Under Observation Schedule Constraints

Ziqi Yang and Roberto Tron

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9340747

***Abstract*— We consider the problem of enhanced security of multi-robot systems to prevent cyber-attackers from taking control of one or more robots in the group. We build upon a recently proposed solution that utilizes the physical measurement capabilities of the robots to perform *introspection*, i.e., detect the malicious actions of compromised agents using other members of the group. In particular, the proposed solution finds multi-agent paths on discrete spaces combined with a set of mutual observations at specific locations to detect robots with significant deviations from the preordained routes.**

In this paper, we develop a planner that works on continuous configuration spaces while also taking into account similar spatio-temporal constraints. In addition, the planner allows for more general tasks that can be formulated as arbitrary smooth cost functions to be specified. The combination of constraints and objectives considered in this paper are not easily handled by popular path planning algorithms (e.g., sampling-based methods), thus we propose a method based on the Alternating Direction Method of Multipliers (ADMM). ADMM is capable of finding locally optimal solutions to problems involving different kinds of objectives and non-convex temporal and spatial constraints, and allows for infeasible initialization. We benchmark our proposed method on multi-agent map exploration with minimum-uncertainty cost function, obstacles, and observation schedule constraints.

***Index Terms*— Path planning, trajectory optimization, map exploration, ADMM**

I. INTRODUCTION

Multi-robot systems have been applied in different scenarios such as field exploration, sensor data collection, surveillance, etc. A prominent example is modern industrial automation where multi-robot systems such as those by Fetch Robotics and Amazon Robotics have been used in commercial settings for a number of years. Another example is given by aerial swarms which have been used in precision agriculture [1], 8 forest fire monitoring [2], and other monitoring or surveillance applications. These multi-robot systems typically employ well-tested solutions for efficient navigation, collision avoidance, and safe interactions with humans (commonly by isolating the robots and humans), In addition, these networked communications based cyber-physical systems are at risk of being hacked. As noted in [3], there are increasingly more high-profile hacking cases; if these incidents were combined with the physical capabilities of robotic systems, the possible adverse consequences could range from exploitation of trade secrets, to production slowdowns, physical property damage, or even human injury [4]. It is therefore important to layer multiple defense mechanisms to enhance the cybersecurity of multi-robot systems.

This work was supported by the National Science Foundation grant NSF CPS 1932162.

In this paper, we build upon a recently proposed *introspection-based* security layer that combines path planning with the agents' physical sensors (each having the ablility to detect other agents) to timely *detect* compromised robots that are not following the pre-planned trajectories. In particular, we want to modify the planned paths for the robots so that they can observe each other frequently enough at key locations so that compromised robots do not have the opportunity to reach forbidden areas (e.g., containing security-sensitive equipment, or human workers) without being detected. For this reason, we introduce the concept of an *observation schedule* that, at a high-level, trades off traditional path-quality measures (e.g., length) for increased security. In our proposed solution, we consider the spatio-temporal constraints induced by the observation schedule at an equal level with other traditional path planning constraints.

A previous work [5] provides an algorithm that solves our problem in a restricted setting: the robots can only move on a subset of a four-connected grid, and the only type of path that can be (indirectly) optimized is the maximum path length. The proposed algorithm is based on a Satisfiability-Modulo-Theory (SMT) solver, for which the complexity scales (in a worst-case scenario) exponentially with the problem size. In this paper, we propose a planner based on Alternating Direction Method of Multipliers (ADMM) that can handle similar spatio-temporal constraints while allowing for continuous configuration spaces and the optimization of arbitrary smooth cost functions.

Paper contributions. We propose a path planning algorithm based on ADMM, a variation of the Augmented Lagrangian Method (ALM, [6]). We choose ADMM for its empirically demonstrated ability to deal with non-convex and non-smooth optimization problems [7]. To the best of our knowledge, our path planning algorithm is the first to handle the spatio-temporal constraints induced by the observation schedule. Additionally, our formulation allows the incorporation of a large variety of types of constraints and the optimization of complex cost functions (e.g., uncertainty in collaborative map estimation). The main limitation of our approach is that it can guarantee only local convergence; however, this can be practically counteracted by using proper initialization techniques, such as using approximate solutions to relaxed or lower-scale versions of the problem, and careful representation of the environment.

We validate our algorithm on a benchmark mapping application on a three robot system where the aim is to minimize the overall uncertainty on a vector field (estimated using Kalman filtering) over a finite time horizon with multiple agents and spatio-temporal constraints (such as

obstacle avoidance and inter-agent observation schedules). While similar tasks have been considered in other works (see Section II for a more detailed review), we are unaware of other planners that can handle the same objective constraints of the same type considered here.

II. Prior Work

Traditional multi-agent path planning problems are usually solved in two ways: centralized approach and distributed approach. Centralized approaches treat the multi-agent as a single agent system which can be solved using single agent planning methods: graph search based algorithms like Dijkstra Algorithms, A^* [8], [9]; sampling based algorithms like $RRT, RRT^*, RRT^\#$ [10]–[12]; and optimal solvers (which will be discussed later). This approach can obtain the optimal result but scales poorly. Distributed approaches treat each agent separately. By splitting the higher dimensional problem into several lower dimensional ones and solving individually, distributed approaches significantly reduce the computation cost. But there is no guarantee of optimality and completeness.

Many recent works have focused on a combination of the two approaches. Biased Cost Pathfinding technique [13] focus on collision prevention by repelling colliding units from the potential collision locations during the planning phase. In [14], the authors present an algorithm based on operator decomposition (OD) and Independence Detection (ID) technique. Agents are first decoupled into non-independent subgroups through ID, then OD computes and update these subgroups in an arbitrary but fixed order.

An area of high interest and activity is the optimization based approaches. These approaches are customizable to the task's specific needs (e.g. maximum surveillance coverage, minimal energy cost) and constraints (e.g. speed limit and avoid obstacles). Potential field methods [15] [16] were introduced for obstacle avoidance. While 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 [17] [18] [19]. Several optimization techniques like MIQP [20] and ADMM [21] have been used to reduce computational complexity and to incorporate more complex non-convex constraints.

Regarding the specific application considered in this paper, there exists some rich literature on multi-agent mapping and exploration problems. The goal of these works are to maximize the coverage of information of an unknown environment [2], [22] by utilizing the information exchanged on the network to solve the problem in a distributed fashion. For example, Rapidly exploring random cycles (RRC) [23], a variant of RRT, is introduced to generate a periodic trajectory for multiple sensing robots to explore a dynamic spatio-temporal field.

III. Notation and Problem Formulation

We consider the problem of finding trajectories for multiple agents that minimize a common objective function while satisfying different types of spatio-temporal constraints.

We denote $q_{ij} \in \mathbb{R}^m$ as the position of agent i at the discrete-time index j, with m representing the dimension of the workspace. For a team of n agents and a task time horizon 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, formally:

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

In this paper, we assume that there exists an underlying slowly-time-varying scalar or vector field $\mathbf{x}$ whose value needs to be estimated at a given number of discrete locations. We then define the cost function $\Phi(\mathbf{q})$ to be an approximation of the minimum uncertainty at any of the given points, and we aim to minimum such maximize uncertainty. This cost function and its gradient are defined more precisely in Section V.

For the set Ω, we consider the following types of constraints:

1) *Start and end locations:* We assume that the start and end locations for each agent, i.e., q_{i0} and q_{iT}, are given.
2) *Velocity constraints:* We enforce approximate constraints on the dynamics of the agents by assuming that they can move a maximum travel distance in any arbitrary direction over discrete time periods. This constraint could be easily modified to use more detailed models.
3) *Convex obstacle constraints:* We model areas that cannot be entered by the agents with convex polygons. These areas can represent physical obstacles, or areas that the agent should not access (e.g., because they contain sensitive information). If the environment contains non-convex obstacles, they can be still modeled using the union of (possibly overlapping) convex obstacles.
4) *Waypoints with flexible deadlines:* We assume that we are given locations that need to be visited by a given agent in a given time window, although the precise instant in that time window can be chosen by the planner.
5) *Introspection constraints:* We assume that two agents can detect each other's presence when they are at a distance of at most d_{max} from each other.

The observation schedule constraints mentioned in the introduction can be modeled as a combination of constraints *4)* and *5)* above (i.e., to require two agents see each other in the vicinity of a given location). Note that these two types of constraints are hard to incorporate for some types of planners, such as those based on potentials or sampling. The precise mathematical description of the path planning constraints listed above is given in Section VI.

IV. Alternating Direction Method of Multipliers

Our proposed path planner is based on the Alternating Direction Method of Multipliers (ADMM) which is a commonly used distributed optimization algorithm [6]. The basic idea behind ADMM 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. Then the problem can be rewritten as:

$$\begin{aligned} \min \quad & \Phi(\mathbf{q}) + \Theta(\mathbf{z}) \\ s.t. \quad & \mathbf{q} - \mathbf{z} = 0 \end{aligned} \tag{2}$$

where Θ is the indicator function of Ω. And the augmented Lagrangian can then be formulated as:

$$L_\rho(\mathbf{q}, \mathbf{z}, \mathbf{u}) = \Phi(\mathbf{q}) + \Theta(\mathbf{z}) + (\rho/2)\|\mathbf{q} - \mathbf{z} + \mathbf{u}\|_2^2 \tag{3}$$

Thus, the search for optimal variables could be separated into two parts: an unconstrained optimization of $\mathbf{q}$, and an optimization of $\mathbf{z}$ with constraint $\mathbf{q}^k - \mathbf{z} = 0$ where $\mathbf{q}^k$ is the result of the first part. This separation allows for a certain amount of violation on the constraints during the process and does not require a basic feasible solution to start the algorithm. The general ADMM iterations is shown:

$$\mathbf{q}^{k+1} := \underset{\mathbf{q}}{\operatorname{argmin}} L_\rho\left(\mathbf{q}, \mathbf{z}^k, \mathbf{u}^k\right) \tag{4a}$$

$$\mathbf{z}^{k+1} := \underset{\mathbf{z}}{\operatorname{argmin}} L_\rho\left(\mathbf{q}^{k+1}, \mathbf{z}, \mathbf{u}^k\right) \tag{4b}$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + \mathbf{r}^k \tag{4c}$$

where $\mathbf{u}$ represents a scaled dual variable, and intuitively accumulates the sum of primal residuals

$$\mathbf{r}^k = \mathbf{q}^{k+1} - \mathbf{z}^{k+1}, \tag{5}$$

which contains the difference between the main variables $\mathbf{q}$ and their copies $\mathbf{z}$.

Remark 1: In practice, the update step of the slack variable $\mathbf{z}$ in (4b) can be interpreted as a Euclidean projection of $\mathbf{q} + \mathbf{u}$ to the constraint set Ω; this is a consequence of the definition of $\Theta(\mathbf{z})$, and the augmentation term contained in the Lagrangian (3).

A. Generalization for easier constraint projections

In the traditional application of ADMM, as noted in Remark 1, the update step for the duplicated variables $\mathbf{z}$ requires a projection to the constraint set Ω. For problems with *convex* constraints, such projection is typically feasible to implement. However, in path planning problems, some constraints are non-convex, rendering the projection step more difficult (due to the presence of multiple local minima in the optimization problem in (2)). To deal with this problem, we propose a minor generalization of the ADMM formulation (2) where we allow $\mathbf{z}$ to replicate an arbitrary function of the main variables $\mathbf{q}$ (instead of being an exact copy):

$$\begin{aligned} \max \quad & \Phi(\mathbf{q}) + \Theta(\mathbf{z}) \\ s.t. \quad & D(\mathbf{q}) - \mathbf{z} = 0 \end{aligned} \tag{6}$$

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

The corresponding update steps are then replaced by:

$$\mathbf{q}^{k+1} := \underset{\mathbf{q}}{\operatorname{argmin}}(\Phi(\mathbf{q}^k) + \frac{\rho}{2}\|D(\mathbf{q}) - \mathbf{z}^k + \mathbf{u}^k\|_2^2) \tag{7a}$$

$$\mathbf{z}^{k+1} := \Pi_\zeta(D(\mathbf{q}^{k+1}) + \mathbf{u}^k) \tag{7b}$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + D(\mathbf{q}^{k+1}) - \mathbf{z}^{k+1}, \tag{7c}$$

where Π_ζ is the new projection to the modified constraint set ζ. Since $D(\mathbf{q})$ is the vertical concatenation of $D_l(\mathbf{q})$, the projection of each set ζ_l is independent for each constraint and can be computed separately. The advantage of this formulation is that we can choose $D(\mathbf{q})$ such that the new constraint set ζ becomes simple to compute; however, the drawback is that we "move the non-convexity" to the primal cost function, i.e., in the update for $\mathbf{q}$ in (7a). Additionally, the function $D(\mathbf{q})$ can be used to select only the subset of the variables on which a constraint depends.

B. Adaptive penalty parameter

The penalty parameter ρ plays a very important role in this method. Intuitively, a small penalty generally allows the solution of the subproblem (4a) to achieve a lower cost while allowing more violation of the primal constraints. Conversely, a large penalty discourages intermediate solutions from violating the constrains, thus increasing the chance that the optimization gets stuck in local optimal solutions. In our application, a static penalty parameter does not work well. Ideally, we would like to have the first iterations of the algorithms focus on obtaining a (possibly global) optimal initial guess (with less regard for the constraints), while gradually bringing back the constraints during the optimization, and modifying the initial guess to eventually satisfy all the constraints. In the ADMM framework, this idea can be formally implemented by updating the penalty parameter based on the primal and dual residuals. More precisely, the penalty parameter is updated according to:

$$\rho^{k+1} = \begin{cases} \tau^{incr}\rho^k & \text{if } \|\mathbf{r}^k\|_2 \le \mu\|\mathbf{s}^k\|_2, \\ \rho^k/\tau^{decr} & \text{if } \|\mathbf{s}^k\|_2 \le \mu\|\mathbf{r}^k\|_2, \\ \rho^k & \text{otherwise.} \end{cases} \tag{8}$$

where $\mathbf{r}^k$ is the primal residual given in (5), μ, $\tau^{incr} > 1$ and $\tau^{decr} > 1$ are adjustable constants used to adjust the penalty (in our case $\mu = 2$, $\tau^{incr} = \tau^{decr} = 1.4$), and $\mathbf{s}^k$ is the dual residual:

$$\mathbf{s}^k = -\rho\left(\mathbf{z}^k - \mathbf{z}^{k-1}\right). \tag{9}$$

C. Complete algorithm

Algorithm 1 is used to solve the generalized problem in (6) with dynamic penalty updates. For convergence proofs regarding this algorithm, see [6].

Algorithm 1 ADMM iteration for problem 2

Initialize $\mathbf{q}^0, \mathbf{z}^0, \mathbf{u}^0$
while $r^k \neq 0$, $s^k \neq 0$ **do**
  $\mathbf{q}^{k+1} := \operatorname{argmin}_\mathbf{q}(\Phi(\mathbf{q}^k) + \frac{\rho^k}{2}\|\mathbf{q} - \mathbf{z}^k + \mathbf{u}^k\|_2^2)$
  $\mathbf{z}^{k+1} := \Pi_\Omega(\mathbf{q}^{k+1} + \mathbf{u}^k)$
  $\mathbf{u}^{k+1} := \mathbf{u}^k + \mathbf{q}^{k+1} - \mathbf{z}^{k+1}$
  penalty update (8)
end while

D. Infeasibility detection

In some situations, ADMM might be unable to find a feasible solution. It is important that we are able to detect when the problem is infeasible, and ideally identify which constraints are the cause. According to [24], this can be done by defining

$$\delta \mathbf{u} = \lim_{k \to \infty} \frac{1}{k} \mathbf{u}^k. \tag{10}$$

The problem is infeasible if $\delta \mathbf{u} \neq 0$. And since each element of $\mathbf{u}$ independently represents the primal residual status of a constraint, the problematic constraints corresponds with the u such that $\delta \mathbf{u} \neq 0$.

V. Minimum Uncertainty Mapping

In this section, we focus on a particular choice of objective function $\Phi(\mathbf{q})$, in the context of a mapping and estimation applications, for the optimization problem (1) .

To explore unknown spatially-distributed fields, autonomous agents are usually required to traverse the entire unknown environment while collecting sensory data to estimate a corresponding map (which we consider to be embodied by a vector field). If the quantity under observation is time-varying, then it is important to consider not only *where* the agents will navigate, but also *when*.

We model the map as a set of locations of interests $\{c_l\}_{l=1}^{n_{loc}}$ arranged on a regular grid, where c_l is represented as the coordinate of the locations (this choice is the most natural in mapping applications, but our proposed method is actually applicable to any arbitrary arrangement of locations). At each location c_l, we assume that there is a corresponding slowly-time-varying quantity $x_{lj} \in \mathbb{R}^{n_x}$ where j represents a discrete-time index. We denote $\mathbf{x}_j \in \mathbb{R}^{n_{loc} n_x}$ as the aggregate state of the quantities of interest at all locations and at time j.

We model the evolution of $\mathbf{x}_j$ as a Gaussian random walk; in addition, we assume that each agent i obtains a measurement $\mathbf{z}_{ij}$ for every location in the environment at every time instant j, but that the confidence in these measurements quickly diminsh with the distance between the location c_l and the agent position q_{ij}. More in detail, we assume the following model:

$$\mathbf{x}_j = \mathbf{x}_{j-1} + \mathbf{v}_{j-1}, \tag{11}$$

$$\mathbf{z}_{ij} = \mathbf{x}_j + \mathbf{w}_j(q_{ij}). \tag{12}$$

where $\mathbf{v}_j$ and $\mathbf{w}_j$ are noise terms distributed according to, respectively, $\mathcal{N}(0, Q_j)$ and $\mathcal{N}(0, R_j(q_{ij}))$; we assume that model noise and measurement errors at different locations and across robots are uncorrelated (and hence independent, since they have Gaussian distribution), so that Q_j and R_j are diagonal; as already mentioned, the covariance matrix R_j depends on the agent location (small when near, and very high when far, the details for this part of the model are introduced below).

We formulate the estimation problem using Kalman Filters (KFs, [25]) which also provides a straightforward way to quantify the uncertainty in the map model. In particular, for our purposes, we are interested in finding paths that best reconstruct the field $\mathbf{x}$, i.e., that achieve the minimum uncertainty; therefore, we use the fact that the uncertainty of the estimates for KFs does not depend on the actual measurements $\mathbf{z}_{ij}$. Hence, we can ignore the estimates of each state $\mathbf{x}_j$, and instead focus on its estimated covariance P_j. Following (11), the KF covariance iteration for P_j can be written as:

$$\begin{aligned} P_j^f &= P_{j-1} + Q_{j-1}, \\ K_j &= P_j^f \left(P_j^f + R_j(\mathbf{q}) \right)^{-1}, \\ P_j &= (I - K_j) P_j^f. \end{aligned} \tag{13}$$

For our application, however, it is more convenient to work with the information form of the KF since we would like to model states and measurements as having zero information (i.e., infinite variance): as in the case of the initial field value x_0; and of measurements of locations that are outside of the agent's sensor footprint. Hence, we replace the covariance matrix P_j with its inverse, the information matrix Y_j:

$$Y_j = P_j^{-1}, \quad Y_j^f \quad = (P_j^f)^{-1}. \tag{14}$$

The information matrix can then be initialized as zero ($Y_0 = 0$) to model the fact that we do not have any a priori information on the field.

From (13), the information matrix update step can be written as:

$$Y_j = (I + Y_{j-1} Q_{j-1})^{-1} Y_{j-1} + I_j(\mathbf{q}) \tag{15}$$

$$I_j(\mathbf{q}) = R_j(\mathbf{q})^{-1} \tag{16}$$

where $I_j \in \mathbb{R}^{nm}$ is the inverse of the measurement covariance matrix which is a function of $\mathbf{q}$. We formulated this function using a Gaussian radial basis function, specifically:

$$[I_j]_{ll} = \sum_{i=1}^{n} K \exp\left(-\frac{\|c_l - q_{ij}\|^2}{2\sigma_c^2} \right), \quad l \in [1, \ldots, nm] \tag{17}$$

where c_l is the coordinate of the field restoration location that needs to be calculated, and σ_c depends on the radius and accuracy of the robot's sensor.

The derivative of Y_j can be computed as:

$$\frac{\partial Y_j}{\partial q_{ij}} = -\frac{\left(Q^T \frac{\partial Y_{j-1}}{\partial q} \right) \left(Q^T \frac{\partial Y_{j-1}}{\partial q} \right)^{\mathrm{T}}}{(I + Y_{j-1} Q_{j-1})(I + Y_{j-1} Q_{j-1})^{\mathrm{T}}} + \frac{\partial I_j}{\partial q_{ij}}. \tag{18}$$

Note that (18) is recursively defined starting with $\frac{\partial Y_0}{\partial q_{ij}} = 0$.

We then formulate the map exploration problem with the goal of finding the trajectory $\mathbf{q}$ that maximizes the minimum information along any of the directions Y_T where T is the task time horizon:

$$\begin{aligned} \underset{\mathbf{q}}{\operatorname{argmax}} \quad & \operatorname{softmin}\left(\operatorname{diag}\left(Y_T(\mathbf{q})\right)\right) \\ s.t. \quad & \forall i \in [0, T], \forall j \in N, \\ & \mathbf{q} \in \Omega, \end{aligned} \tag{19}$$

where the function $\operatorname{softmin}$ is defined as a smooth approximation of the $\min$ function, defined as:

$$\operatorname{softmin}(a_1, \ldots, a_n) = \log(e^{a_1} + \cdots + e^{a_n}). \tag{20}$$

VI. Path Planning Constraints

In this section, we provide mathematical descriptions for a variety of constraints. As introduced in Section IV-A, constraints are decoupled and reformulated as mapping functions $D(\mathbf{q}) = \mathbf{z}$ and constrained sets ζ. Also, instead of solving a possibly non-convex optimization problem in (4b), projections to the constrained sets $\Pi_\zeta(\mathbf{z})$ will do the work.

A. Start and end locations

For each agent, we fix the starting and end locations, i.e., q_{i0} and q_{iT}. These are simply enforced by removing them from the optimization variables. However, for applications where the agents are modeled with more complex dynamics, we would need to include q_{iT} as a variable, and add the corresponding linear constraint to the optimization.

B. Velocity constraint

Since we are using a discrete formulation, the velocity of an agent can be approximated by taking the difference between adjacent waypoints on the trajectory. We then define the function $D(\mathbf{q})$ to return the velocity vectors for the i-th agent at time step j:

$$D_{ij}(\mathbf{q}) = q_{ij} - q_{i(j-1)}, \quad j \in \{1, \dots, T\} \tag{21}$$

and the constraint set is:

$$\zeta_{ij} = \big\{ z \mid \|z\| \leq v_{max} \big\}. \tag{22}$$

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

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

C. Convex obstacles

In our motivating scenarios, there are areas that the agents should not visit, because they represent forbidden zones or obstacles to avoid. We model these zones using convex polygons defined by several hyperplanes and having the normal vectors of these hyperplanes pointing outside the obstacles. We enforce the constraints at each discrete time step q_{ij} (enforcement of the constraints between these points can be achieved by making the obstacles sightly bigger than its actual size). For a convex area, waypoints stays unchanged it they are outside the zone, and if the waypoints are inside the zone, the constraint function $D(\mathbf{q})$ returns the least negative distance from the waypoints to the zone defined by hyperplanes which is represented as:

$$D_i(\mathbf{q}) = \begin{bmatrix} d_{11} \\ \vdots \\ d_{nm} \end{bmatrix} \tag{24}$$

where

$$d_{ij} = \max(\min(d_{ijk}, 0)), \tag{25}$$

$$d_{ijk} = p_{ij}^{\mathrm{T}} n_k - m_k, \tag{26}$$

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

$$\zeta = \{ z \mid z = 0 \}, \tag{27}$$

with a projection function:

$$\Pi_\zeta(z) = 0. \tag{28}$$

The motivating idea for (24) and (28) is to find waypoints inside the zone and project them to the closest boundary. As mentioned before, non-convex obstacles can be handled by the union of (possibly overlapping) convex obstacles.

D. Waypoints with flexible deadlines

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

$$D_i(\mathbf{q}) = \min_{i \in \{1 \dots n\}, j \in \{t_1, \dots, t_2 - 1\}} \big(\mathrm{dist}(p, \overrightarrow{q_{ij} q_{i(j+1)}}) \big) \tag{29}$$

with the constraint set:

$$\zeta = \{ z \mid z < z_{max} \} \tag{30}$$

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

$$\Pi_\zeta(z) = \min(z, z_{max}). \tag{31}$$

Note that (30) and (31) are equivalent to (22) and (23), except for the fact that the quantity in (29) is always a positive scalar.

E. Introspection constraint

In this type of constraint, the agents are required to be in physical proximity of each other for some time instant j belonging to a given time window $[t_1, t_2]$ (e.g., to inspect each other, or to exchange data). We write the function for the constraint as:

$$D(\mathbf{q}) = \min_{j \in [t_1, t_2]} \| \overrightarrow{q_{aj} q_{bj}} \|, \tag{32}$$

where a, b are the indices of the pair of agents required for a mutual inspection. With this definition, the constraint set and the projection operator are the same as (22) and (23).

F. Implementation as a Matlab toolbox

We implemented our planner in Matlab. We structured our code in a modular way, so that different sets of constraints among the types already described in this section can be easily incorporated; new constraints can be added by simply specifying the corresponding function $D(\mathbf{q})$, its Jacobian (to enable the use of gradients during minimization), and the projection operator Π_ζ. For the primal unconstrained optimization step (7a), we used the standard Matlab routine `fminunc`; nonetheless, our framework also supports custom solvers tailored for the special structure of the given problem (for instance, we have written a solver for quadratic problems). We will release our code to the community as an open source package.

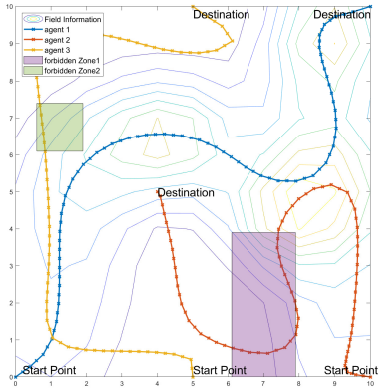
VII. Results

In this section, we apply our ADMM path planning algorithm to an instance of a map exploration problem, 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. We set the sensor accuracy for each agent in (17) to $\sigma_c = 1$; And assuming that the robots can meaningfully collect information on the underlying vector field for data locations no further than a radius of $3m$. The maximum distance for the velocity constraints v_{max} is set to $0.5m/dt$.

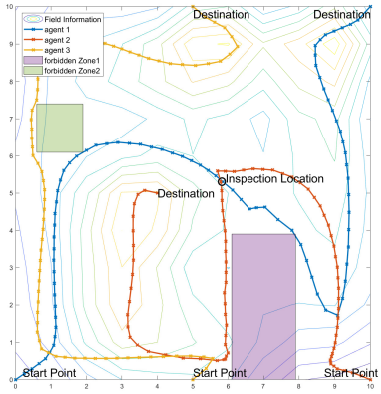
A. Simulations

Our baseline test is one that only considers the velocity constraint, the result can be seen in Figure 1a. The contours show the estimated accuracy for the map which is the result of the information matrix (the larger the better). The elements in the resulting entries of the information matrix for this test has a max of 5.5 and a min of 1.8. Note that the forbidden zones (obstacles) were not included as constraints which explains the zone violation by the agents. The trajectories are initialized as straight lines with the agents moving at uniform speed from their start to their goal locations. After running our proposed method, the trajectories assume boustrophedon-like patterns. These patterns have been traditionally proposed for coverage problems, but here they appear as a byproduct of our cost function while taking into account the spatio-temporal behavior of the uncertainty; for instance, the trajectories of agent 1 and agent 2 appear to be very close, which would be suboptimal in terms of minimizing the maximum uncertainty; in fact, agent 1 reaches the point closest to agent 2's path well after agent 2 has left the corresponding point on its trajectory; in a sense, agent 1 and agent 2 automatically take turns to cover that location.

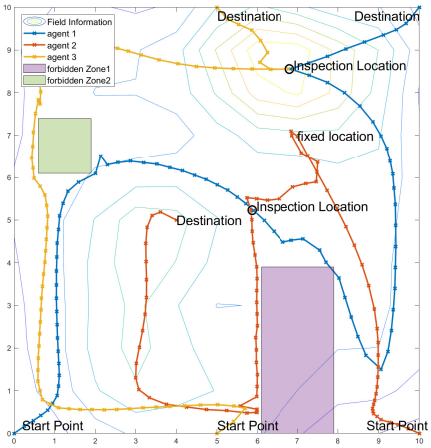
We use the solution of the baseline test to initialize another test with some additional constraints: two forbidden zones (shown as colored boxes in Figure 1), and one inspection constraint between robot 1 and robot 2 at time 40. Note that the initial trajectories, from the baseline test, is infeasible since the agents cross the forbidden zones (see Figure 1a); our algorithm is capable of handling the new constraints and returning a feasible solution, as shown in Figure 1b. Note



(a) Baseline test with only velocity constraint



(b) Forbidden zone and inspection constraints added to the baseline test



(c) Additional constraints added on top of 1b

Fig. 1: Resulting trajectories for map exploration task under different constraints.

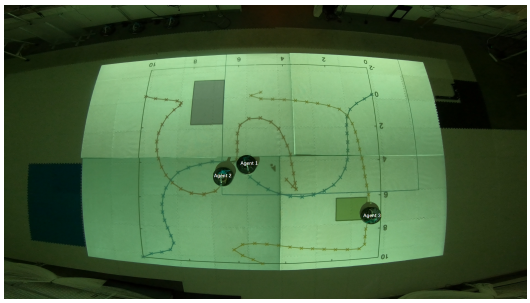


Fig. 2: Experiment of a team of three robots where agent 1 and agent 2 meet at the pre specified time and location.

that the algorithm corrects the trajectory for agent 2 from near the right edge of the purple obstacle to the left of the obstacle. Resulting entries of the information matrix for this case have a max of 7.2 and a min of 1.4.

We further increased the number of constraints in a third test: we added a waypoint constraint for the location at $[7, 7]$ (at any point in the mission), and robot 1 and robot 3 were asked to meet at about time 70. The result can be seen in Figure 1c with resulting entries of the information matrix having a max of 10.0 and a min of 1.3.

B. Experiments

The simulation result in the second test case, Figure 1b, has been selected for experimental validation using a team of three real robots. The goal of this experiment is to see if the plans can be followed despite the mismatch between the approximated behavior used in our algorithm and the actual robots. Our platform consists of three modified ground robots, iRobot Create2, that are controlled with a Raspberry Pi, and tracked using an OptiTrack Motion Capture System. For visual reference, we use short-throw projectors to visualize the environment and the paths. Each robot is controlled by an embedded PD controller which is fed the sequence of waypoints given by the planner. The full setup is shown in Figure 2. We verified, by inspection, that the robots successfully followed the waypoints designed by our algorithm while meeting the observation constraint. Full results can be viewed in the accompanying video submission.

VIII. Conclusion and future work

In this paper, we have presented a path planning algorithm that can generate optimal trajectories for multi-agent systems while satisfying complex spatio-temporal constraints. This planner can be used to support new methods to enhance the security of networks of robots against malicious take-overs of agents. Our solution is based on an application and modification of the ADMM framework. Simulation and experimental results show that our method, while based on local optimization, is able to obtain solutions for non-trivial planning problems involving cumulative cost functions, waypoints with flexible deadlines, and introspection constraints.

A Matlab implementation of our method will be released as an open-source flexible toolbox. The current toolbox does not scale well for problems involving many agents and longer time horizons, future work is to distribute the primal optimization problem in (7) among each agent using the techniques introduced in [7], [18], [21], and to replace the high-dimension optimization problem with several lower-dimensional ones. In addition, collision avoidance is not considered in the current implementation, but will be introduced in the future.

References

[1] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–330, 2015.

[2] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

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

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

[5] K. Wardega, R. Tron, and W. Li, "Resilience of multi-robot systems to physical masquerade attacks," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 120–125.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," vol. 3, no. 1, pp. 1–122, 2011.

[7] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[8] H. M. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[9] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[10] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[11] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," 2011, pp. 1478–1483.

[12] F. Hauer and P. Tsiotras, "Deformable rapidly-exploring random trees," in *Robotics: Science and Systems*, 2017.

[13] A. Geramifard, P. Chubak, and V. Bulitko, "Biased cost pathfinding." 2006.

[14] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[15] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE transactions on systems, man, and cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[16] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[17] X. Liu and P. Lu, "Solving nonconvex optimal control problems by convex optimization," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 750–765, 2014.

[18] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," *2016 European Control Conference, ECC 2016*, pp. 1580–1585, 2016.

[19] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[20] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 477–483.

[21] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Advances in neural information processing systems*, 2013, pp. 521–529.

[22] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.
[23] X. Lan and M. Schwager, "Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, 2016.
[24] G. Banjac, P. Goulart, B. Stellato, and S. Boyd, "Infeasibility Detection in the Alternating Direction Method of Multipliers for Convex Optimization," *2018 UKACC 12th International Conference on Control, CONTROL 2018*, p. 340, 2018. [Online]. Available: https://doi.org/10.1007/s10957-019-01575-y
[25] B. D. Anderson and J. B. Moore, *Optimal filtering*. Courier Corporation, 2012.