AER1516 Final Project: Investigating Robustness of Multi-Robot Trajectory Planners to Communication Dropout

Nurein Umeya, Federico Pizarro Bejarano, Jaime Fajardo

April 19, 2023

1 Introduction

Efficient and safe multi-robot trajectory generation with online collision avoidance is vital for the operation of swarms of robots, which can be used in a variety of applications including performances [30] and search and rescue [20, 31]. Many approaches to real-world multi-robot trajectory generation exist, but most use the assumption of perfect, instantaneous communication with no dropout (i.e., [22]). In the real-world, not only are there variable communication delays [27, 17], but communication may fail entirely for periods of time [32].

This project will investigate robustifying multi-robot trajectory generation algorithms to communication dropout. First, in Section 2 we will motivate the importance of considering communication failure in multi-agent systems. Then, in Section 3 we will analyze the literature around decision-making and communication architectures, with an emphasis on decentralized architectures and implicit coordination. Then, using the example of the state-of-the-art MPC-based algorithm [22] (described in Section 3.4), we will apply several methods of improving performance when communication fails. By experimenting with different methods of accounting for communication failure in Section 4, we hope to gain insight into the challenge of maintaining safety and performance when communication fails, as well as introducing simple methods which all algorithms can apply. Our results can be found in Section 5.

2 Background

Trajectory generation can be centralized, where one central agent determines the trajectory for all agents, or decentralized, where each agent generates its own trajectory [38]. Centralized architectures use a central computer which has access to all the available information to make decisions and plan globally optimal trajectories for all the agents. However, centralized architectures suffer when communication is unreliable [27], cannot scale to larger teams [38], and present a single point of failure, the central agent [38]. In contrast, in decentralized architectures each robot calculates its own decisions, where coordination with other robots is done through limited communication and approximations of the behaviour of other agents [38]. Decentralized methods can scale to large teams and are robust to failure but lead to suboptimal solutions in general [38].

Decentralized architectures coordinate the agents when there is a conflict, such as task allocation or avoiding collisions. Coordination methods can be roughly divided into explicit and implicit coordination. Explicit coordination has each agent communicate with the others to jointly agree on actions. Similar to centralized approaches, explicit coordination may result in superior performance but is more computationally expensive and fails when communication drops. Implicit coordination does not require direct communication between agents, but instead infers their actions. In real operation it is likely that communication between agents will fail, either dropping intermittently or introducing significant latency and limiting bandwidth [27]. Due to this, an optimal algorithm is one that can make use of communication when it is available, but is still robust to communication failure.

Many decentralized trajectory generation algorithms rely on each agent sharing their current position and future trajectory to coordinate [40, 22, 15]. This can be considered a hybrid of implicit and explicit as the robots require direct communication but the coordination is done by each robot without explicitly agreeing on a decision. The paper we are basing our investigation on [22] uses this method, and thus encourages collision avoidance but does not certify it. Despite the reasonably decentralized approach, this system is still sensitive to communication delays and failure, as the robots will collide with greater probability if the trajectories of their neighbours are inaccurate or inaccessible. To encourage collision avoidance when communication fails, the algorithm needs to be augmented with fully implicit coordination strategies that use previously shared trajectories and positions to coordinate when communication fails.

3 Literature Review

The following analysis of various multi-robot coordination strategies will help in the creation of ones that are more resilient to communication failure. Different assumptions are made by each solution regarding the robot team, the accessibility of communication, and the particular task to be carried out. Walls and other obstructions may completely prevent communication, reduce its range, or make it sporadic [8, 27]. In addition to working when communication is unreliable, the algorithms we are designing ought to be well suited for operation in the real-world.

The decision-making architectures (centralised, decentralized, or hybrid) and the coordination methods (explicit or implicit) were reviewed to understand the field of multi-agent coordination and decision-making and help to find an underlying algorithm to modify as well as strategies to improve the underlying algorithm. In Section 3.1, we first go over the various decision-making architectures. Since centralised architectures are inappropriate for settings with intermittent communication, this section serves merely as an overview. Then, in Section 3.2 and Section 3.3, particular decentralized coordination techniques are discussed. The algorithm that was picked as the foundation for our investigation is outlined in Section 3.4.

3.1 Decision-Making Architectures

There are two main approaches to coordinating teams of robots: centralized and decentralized. Additionally, several approaches fall somewhere in between and have been labelled hybrid architectures. There is a spectrum from completely centralized to completely decentralized, and due to unreliable communication, we will focus on highly decentralized approaches. See Figure 1 for a summary of the different architectures.

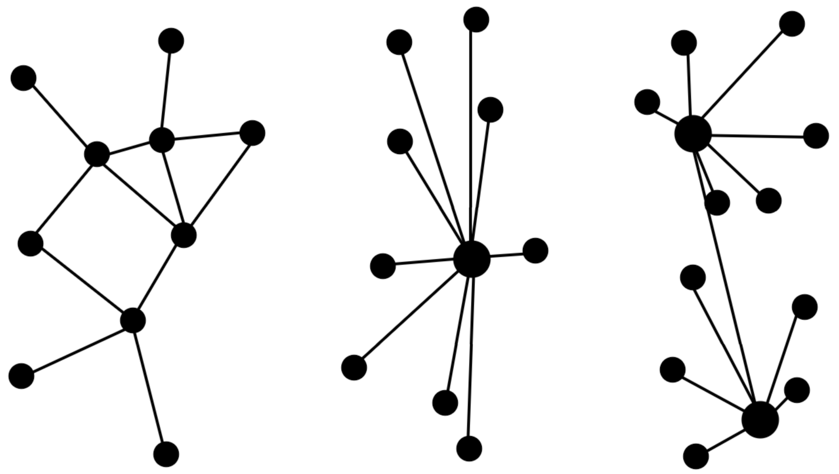


Figure 1: The three different multi-robot architectures. Decentralized (left) where each robot makes it's own decisions and communicates with neighbours, centralized (centre) where all robots communicate to a central agent, and a hierarchical hybrid architecture (right) where the team is split into smaller centralized teams.

3.1.1 Centralized

Centralized architecture involves using a central agent, either a static computer or one of the robots in the team, to make decisions for all the robots in the team [38]. This central agent has access to all the robots' information and all available environment data. Since the central agent has access to all available data, it is capable of planning globally optimal solutions [38]. Centralized architectures have been applied to various problems such as task allocation [18], path planning [23], including for high-dimensional drone teams [2].

Centralized architectures generally suffer from ineffectual scaling to larger teams and vulnerability to issues in the central agent, compounded by unreliable communication [38, 27]. If each robot in the team sends all sensor readings to the central agent, it would require prohibitively large bandwidth for a large team. Additionally, acting upon all the data from a large team would require very large computational resources. Lag in communications or disruptions would greatly affect the entire team, freezing it completely when it cannot receive orders from the central agent. In real situations, communication is usually unreliable [27], and so it is necessary to use methods less reliant on communication and more robust to failures in robot team members.

Centralized approaches can also be calculated offline and then implemented on real robots with no further communication [2, 21]. These approaches would not have the identified weaknesses of centralized approaches, as there is no single point of failure or reliance on communication, but are incapable of reacting online to disturbances. In real-world situations where obstacles are dynamic and disturbances must be accounted for, offline approaches are infeasible. Thus, we will focus exclusively on online approaches.

3.1.2 Decentralized

Decentralized architectures enable for independent decision-making by each robot, with modifications based on observations and collaboration with other robots [38]. It often simply entails passing on information to nearby robots, with little direct communication between robots. The neighbourhood taken into account varies per algorithm and may also be determined by the range of potential communication, as long-range communication may be impractical due to bandwidth constraints [27]. Decentralized approaches produce sub-optimal results since each agent only has access to local knowledge, but are more robust to failure [38]. When considering unreliable communication, decentralized architectures are required due to their low dependency on communication and resilience to failure.

Although communication between robots is not prohibited by decentralized architectures as long as each robot plots its own path, decentralized architectures' key advantages are negated by algorithms that heavily rely on inter-robot communication. In this work, algorithms that call for each robot to be connected to every other robot and so make decisions based on global knowledge are classified as hybrid architectures (discussed in Section 3.1.3).

Some algorithms, such as [29, 39], do not use communication; instead, each robot uses information gathered from interaction with the environment, making the system very robust. Unfortunately, these solutions are less optimal. Systems that communicate to adjacent robots to jointly plan actions [4, 37, 5] are considered decentralized because they only require communication with adjacent robots. However this also may lead to excessive communication and computationally expensive decision-making in large teams where each agent may have many adjacent neighbors. [4, 35, 38] have shown that a hybrid of explicit and implicit coordination is effecting at reducing communication load while maintaining performance. Lastly, there are systems that can work with any level of communication and remain effective by making use of implicit coordination [9, 24], which is the goal of this project.

3.1.3 Hybrid

Classic examples of hybrid architectures include those that separate the robot team into smaller, centralised teams, each with a team captain. Because there are two or more levels of authority inside the team, these are known as hierarchical architectures [38].

Hybrid architectures include decentralized designs that call for constant team contact, such as those in [25, 33, 12]. Even when connectivity is unstable, certain algorithms are explicitly created to keep this global communication network operational [11, 28]. Other algorithms circumvent the communication limitations by arranging for all the robots to rendezvous frequently and collectively plan every goal [16]. Although each robot continues to make its own decisions, the fundamental advantages of decreased communication loads and decentralized decision-making are no longer applicable. Furthermore, decisions made by one robot will have a major impact on those made by all other robots, not simply those that are nearby. Algorithms that need global communication will be regarded as hybrid architectures for these reasons.

3.2 Explicit Coordination

There are two primary methods—explicit and implicit—for coordinating two or more robots in a decentralized architecture. Explicit coordination entails direct communication with other robots and joint task decision-making [38]. Implicit coordination is the process of inferring another robot's plans by either little communication or observation [38].

Although communication is a component of explicit coordination, since it is still a decentralized strategy it uses far less communication than what is required for centralized systems [38]. Instead, it usually requires local communication. There are several prominent outliers that call for global communication and should be categorised as hybrid architectures [25, 33]. Implicit coordination is useful because it needs substantially less communication than explicit coordination, which is preferable since communication is unreliable [38].

Due to the continuous nature of trajectories, explicit coordination is typically better appropriate for task-allocation or frontier-based exploration activities rather than trajectory generation. The state space can be discretized, nevertheless, to enable explicit coordination, as in [15], where robots cooperate to optimize a policy while communicating and approximate each other's behaviours when not in communication.

3.2.1 Negotiation

Robots must clearly agree on the choices for the local group in order for explicit coordination to take place. The robots compute the reward and cost of each assignment and distribute them such that the cost to the team is kept to a minimum. This is best accomplished through bargaining. This approach to global optimality may necessitate heavy computation and communication loads because it is necessary to determine how much it will cost each robot to complete a task [37]. A classic example of multi-robot exploration, the M+ algorithm [4], reduces the number of tasks by only considering optimality in the next time increment.

3.2.2 Market Strategies

Several market economics-based solutions have been developed to lessen the communication and computation requirements of comprehensive consensus and optimality analysis. These include trading techniques [37], auction or bidding tactics [5, 10], and so on. In these tactics, the robots frequently play many roles, occasionally serving as bidders and auctioneers. These techniques can be regarded as hybrid architectures since they rely on judgements made by other robots and produce hierarchies, although momentarily.

In the well-known Gerkey and Mataric auction approach [10], an auctioneer robot would announce a task and the metric defining a robot's fitness to do that assignment. Every robot that can possibly complete the assignment assesses its fitness and gives it to the auctioneer as a "bid." The auctioneer selects the winner in the end and keeps track of their performance on the job. The method advances prior research though still needs strong communication [37]. Trade strategies [37] try to reduce the amount of communication needed by letting buyers request tasks rather than waiting for auctions, and sellers choose winners based on messages received. The buyers greedily choose assignments [37].

3.3 Implicit Coordination

Implicit coordination is the process through which nearby robots may deduce one another's intentions via little verbal or observational cooperation [38]. Instead than explicitly assigning goals to each robot, which may take a lot of time and involve heavy computing and communication burdens, each robot makes predictions about what the others could do, resulting in emergent behaviour [38].

Implicit coordination makes task selection decisions based on knowledge of the movements and locations of other robots in order to predict their intentions. The position and desired trajectory of each robot is often shared to help ascertain their behaviour. A robot can use this information to develop paths that won't cross over with others. Each robot will choose its own aim while taking into consideration the goals of others in a variety of methods, such as minimising deviations from desired formations or preserving separating hyperplanes between each robot [22], staying within their own Voronoi cell [40], or minimizing deviations from intended formations [32].

Different types of information can be shared by these algorithms, such as only locations [40], position and future trajectories [22], or directly shared sensor and odometer readings [9].

3.3.1 Best Fit Task

The aim of an algorithm may be chosen by determining which frontier or goal it is better capable of attaining than other robots. In his landmark work, Yamauchi [36] describes a method in which robots collaborate to create shared maps by sharing perceptual information, in an effort to disperse frontiers for exploration. However, this tactic of implicit cooperation allows for repeated effort by other robots [3]. By deploying robots optimally over the map rather than just towards the nearest border, [3] enhances frontier allocation. A similar concept is explored for general jobs rather than borders in citation [34]. The best suited job to do at each time step must be determined, but this requires expensive computational work.

3.3.2 Approximation Methods

Many algorithms employ estimates of the other robots to avoid collisions and complete tasks rather than determining the best task or frontier for each robot to pursue. These techniques gain from more responsiveness and less computation.

Repelling robots from other robots and attracting them to certain areas, like frontiers, is a straightforward strategy. The robot can 'flow' to low potentials near boundaries and far from other robots utilising algorithms like [19, 1], which directly do this using potential fields superimposed on the surroundings. The value iteration approach is used in [24] to solve Markov decision processes, adding positive rewards to frontiers and negative rewards to regions near other robots. Lastly, [26] uses a particle filter to coordinate a group of robots searching for a target by distributing random particles among them so that each robot may generate a local particle distribution of the target's position. These techniques make use of an appropriate surrogate problem, such as maximising reward or minimising potential, rather than explicitly calculating an optimal action.

Implicit coordination requires that each robot be aware of the other robots' intentions. Many of the analysed algorithms only operate on homogeneous teams, which presuppose identical robots making decisions based on the same algorithm. Multi-robot teams, on the other hand, could consist of disparate robots, such as rovers and drones. These varied teams can need more sophisticated coordination algorithms [38], like [35], which learns the team members' behaviours. After developing a model of each member's conduct, it may make plans based on what its team members are most likely to do.

3.3.3 Biomimicry

Biomimicry, which often imitates swarm insect communication, is a less popular strategy for coordination. In most cases, these techniques employ indirect forms of communication instead of direct electronic communication. One approach is modelled after bees' complicated aerial motions, or "waggle dance," which they use to communicate with one another [7]. The study discusses several messages that UAVs can transmit by projecting shapes into the air. This is helpful when a group of UAVs needs to exchange information but are unable to do so directly. Similar to how ants use pheromones to walk roads and leave messages, other methods employ chemicals like [6] to construct trails that other robots may use for various purposes. Coordination through manipulating the environment is known as stigmergy [7, 6].

However, because these gestures and implicit communication techniques can only convey pre-programmed signals rather than paths or objectives, biomimicry has limited applicability. Additionally, agents frequently need additional sensing tools, like as chemical sensors to identify synthetic pheromones or computer vision software and hardware to identify waggle dance motions.

3.4 Candidate Algorithm

To investigate making an algorithm robust to unreliable communication, an algorithm that assumes perfect communication was required. The algorithm that was chosen is "Online Trajectory Generation with Distributed Model Predictive Control for Multi-Robot Motion Planning" [22], and is referred to as the candidate or chosen algorithm in this project. This algorithm was chosen for the three reasons. Firstly, it is a state-of-the-art algorithm that was the first algorithm to allow for swarms of up to 20 drones to be flown online in the real world, and is a realistic algorithm for deployment in the real world. Second, it demonstrated the efficacy of its collision avoidance technique, comparing its approach to the well-studied buffered Voronoi cell technique [40]. Finally, this paper has open-source code that makes working with the algorithm feasible in the short time-frame of this project, as programming the MPC from scratch would be practically infeasible due to the complexity of the algorithm and the large number of tuning parameters.

This section will briefly go over the important aspects of this algorithm, going into more detail into aspects relevant to this project. For more details, consult the paper.

3.4.1 General Overview

The chosen algorithm is a decentralized trajectory generation algorithm designed for swarms of micro air vehicles (MAVs), although it can be generalized to other vehicles. It assumes the drones are equipped with position controllers, and thus the generated trajectories are directly sent to the drones as position set-points. The drones are assumed to instantaneously and perfectly communicate their current position and intended future trajectories up to a horizon H. The task that is used to test the approach is the point-to-point transition task where the drones begin at starting positions and must reach goal positions as rapidly as possible without colliding.

The algorithm is optimally suited for real-world operation in continuous space as it uses a distributed model predictive control (MPC) structure that solves a quadratic program (QP) at every iteration. The MPC ensures that constraints on the trajectory (such as continuity and smoothness) are enforced, as well as state constraints (such as remaining in a set area), while still minimizing energy expenditure and distance to the goal. Collision avoidance is done through a novel slackened constraint that maintains a hyperplane separation between drones that are close to colliding. These details are described below, as well as its performance compared to the buffered Voronoi cell approach.

3.4.2 Trajectory Parameterization

The trajectories in the chosen algorithm are parameterized as a series of Bézier curves. Bézier curves have several useful properties that make them an excellent choice [13, 14]. Firstly, the curves represent a continuous trajectory using a small number of control points. The control points of the intended trajectory of the i^{th} drone are denoted $\boldsymbol{\mathcal{U}}_i \in \mathbb{R}^{3l(p+1)}$, where l is the number of Bézier curves and p is the degree of the curves. The control points $\boldsymbol{\mathcal{U}}_i$ are the optimization variables of the $i-$th MPC problem. Both sampling the curve and calculating the derivatives can be done analytically using linear combinations of the control points, which is important in ensuring the final MPC optimization is quadratic. Continuity of the curves up to a certain derivative is guaranteed by ensuring the curves are connected together properly, which is done by an equality constraint $\mathbf{A}_{\text{eq}}\boldsymbol{\mathcal{U}}_i = \mathbf{b}_{\text{eq}}$. Additionally, smoothness up to arbitrary derivatives can also be ensured by box constraining samples of the trajectory and their derivatives to be within specified bounds using an inequality constraint, $\mathbf{A}_{\text{ineq}}\boldsymbol{\mathcal{U}}_i \leq \mathbf{b}_{\text{ineq}}$. This linear inequality conservatively bounds the trajectory and its derivatives in boxes [1], but is vital to maintain the QP structure. More detail on determining these matrices can be found in the paper.

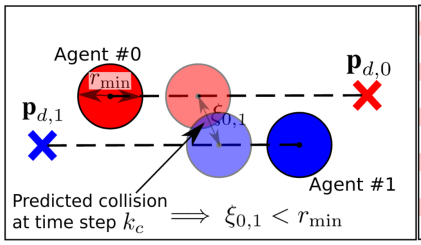
3.4.3 Collision Avoidance

The collision avoidance constraint is

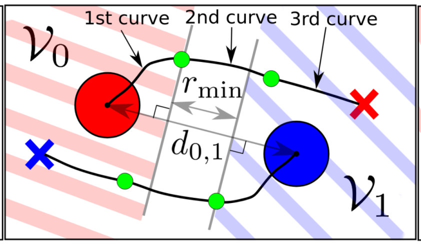
$$\|\boldsymbol{\Theta}^{-1}(\mathbf{p}_i[k_t] - \mathbf{p}_j[k_t])\|_2 \geq r_{\min}, \tag{1}$$

where $r_{\min}$ is the minimum distance the drones must be from one another, $\mathbf{p}_j[k_t]$ is the position of the $j-$th done at time step k_t, and $\boldsymbol{\Theta}$ is a scaling matrix to maintain ellipsoidal safety bounds around each obstacle. This is visualized in Figure 2a.

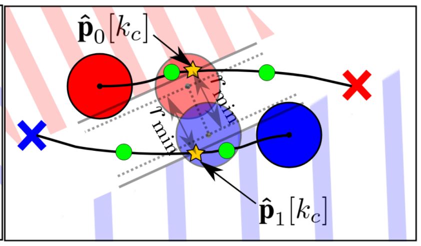
Collision avoidance is done using an 'on-demand' approach, first proposed in [21]. It is defined in two ways: state-space and input-space. In the state-space approach, a model of the system is needed to predict



(a) Upcoming predicted collision



(b) Buffered Voronoi cell approach



(c) On-demand approach

Figure 2: Visualization of the predicted collision, and the BVC and on-demand approaches to avoiding this collision. These figures are originally presented in [21], and are duplicated here for clarity.

the future states of the system given the inputs, and this is used to determine future collisions. The input-space approach assumes that the system tracks the desired trajectory sufficiently well, and thus looks for collisions along the commanded trajectory. The input-space approach does not need a model of the system and performed better in the experiments of that paper, and thus it is the approach we will use in this project.

The on-demand approach determines the first time step where Equation 1 will be violated, and then enforces the constraint up to that time step. It does this using a first-order approximation, which constrains the drones to remain in their separating hyperplane. For feasibility, this constraint is slackened. For more details, see [21]. This is visualized in Figure 2c.

The paper also defines a buffered Voronoi cell approach based on the work in [40]. The Voronoi cell of the $i-$th drone is defined as:

$$\mathcal{V}_i = \left\{ \mathbf{p} \in \mathbb{R}^n \mid \frac{\mathbf{\Theta}(\mathbf{p}_i - \mathbf{p}_j)^\intercal(\mathbf{p} - \mathbf{p}_i)}{d_{i,j}} \geq \frac{r_{\min} - d_{i,j}}{2} \right\}, \forall j \neq i, \tag{2}$$

where $d_{i,j} = \|\mathbf{\Theta}^{-1}(\mathbf{p}_i - \mathbf{p}_j)\|_2$, and $\mathbf{p}_i$ is the position of the $i-$th drone at time step k_t.

Using Equation 2, we can define a linear constraint to ensure collision avoidance. First, let us define $\mathcal{P}_{i,1}$ as the control points associated with the i^{th} drone's first Bézier curve. By constraining these points to be within their cell, $\mathcal{P}_{i,1} \in \mathcal{V}_i$, we ensure safety. This is visualized in Figure 2c.

Both of these approaches can be expressed as a linear inequality, $\mathbf{A}_{\text{coll}}\boldsymbol{\mathcal{U}}_i \leq \mathbf{b}_{\text{coll}}$.

3.4.4 Final MPC Structure

Now that the trajectory parameterization, constraints, and collision avoidance have been defined, we can build the QP that forms the MPC problem. The MPC follows the following structure. At each time step, the QP is solved for the optimal control points defining the optimal Bézier curves for the next horizon H time steps. The first input, sampled from these control points, is applied to the system, then the QP is reinitialized and solved again. This is repeated for the entire trajectory. The QP is structured as

$$\begin{aligned}
\min_{\boldsymbol{\mathcal{U}}_i, \varepsilon_{i,j}} \quad & J_{i,\text{error}} + J_{i,\text{energy}} + J_{i,\text{violation}} && \text{(3a)}\\
\text{s.t.} \quad & \mathbf{A}_{\text{eq}}\boldsymbol{\mathcal{U}}_i = \mathbf{b}_{\text{eq}} && \text{(3b)}\\
& \mathbf{A}_{\text{ineq}}\boldsymbol{\mathcal{U}}_i \leq \mathbf{b}_{\text{ineq}} && \text{(3c)}\\
& \mathbf{A}_{\text{coll}}\boldsymbol{\mathcal{U}}_i \leq \mathbf{b}_{\text{coll}} && \text{(3d)}\\
& \varepsilon_{i,j} \leq 0, && \text{(3e)}
\end{aligned}$$

where $\varepsilon_{i,j}$ is the slack variable associated with the possible collision between the $i-$th and $j-$th drones, and $J_{i,\text{error}}$, $J_{i,\text{energy}}$, and $J_{i,\text{violation}}$ are the cost functions associated with the distance to the goal, the higher derivatives of the trajectory, and the size of the slack variable respectively. $J_{i,\text{error}}$ is simply a sum of the quadratic distances between the drone at each future time step and the goal, $J_{i,\text{energy}}$ penalizes the magnitude of the higher derivatives of the trajectory, and $J_{i,\text{violation}}$ is the sum of a quadratic and linear penalty on the slack variables $\varepsilon_{i,j}$.

For this project, this QP will be modified in various ways to account for communication failure, including adding terms to the objective and modifying the collision constraints.

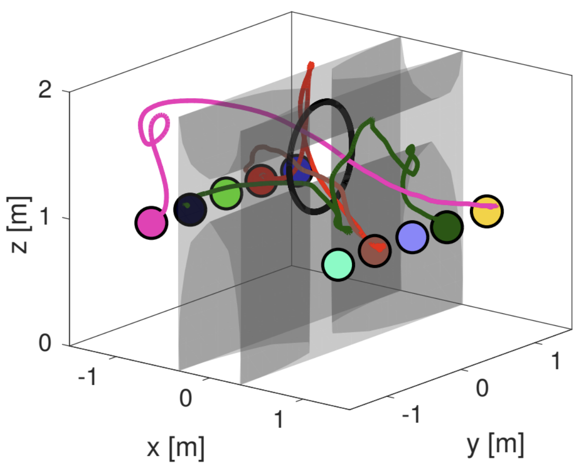


Figure 3: The 10-drone point-to-point transition problem showcasing the ability of the chosen solution. The drones must pass through a hula-hoop (the black ring). The coloured circles are the starting positions of the drones, and the associated lines are the executed trajectories of 4 of the drones. The drones were Crazyflie quadrotors.

3.4.5 Results

The on-demand approach resulted in significant decreases in total mission time (less than half the time of BVC), and significantly improved success probability, especially as the number of drones increased. More impressive, and the reason this paper was chosen, the paper demonstrated its ability to run real experiments with swarms of up to 20 drones. The final experiment was a 10 drone point-to-point transition problem where 5 drones started on either side of a hula hoop suspended in the air. The drones needed to go through the hula hoop to their goal positions on the other side. Due to the significantly constricted corridor of the hula hoop, and the large number of drones, this was an impressive achievement, and can be seen in Figure 3. Further details can be found in [22].

4 Implementation

4.1 Tested Strategies

When developing strategies for allowing the chosen algorithm to function with unreliable communication, it became apparent that there is an trade-off between safety and performance. The underlying algorithm itself does not certify safety, as it has a limited MPC horizon, only communicates with nearby drones, and has a slackened collision constraint. Thus, the underlying algorithm already has a trade-off between safety and performance balanced by several tuning parameters. Similarly, when communication fails, safety can only be reasonably guaranteed for the few iterations until any expected collision, or until the end of the MPC horizon. After this, the uncertainty in the positions of other drones, both ones with known (if outdated) trajectories and ones who were outside the original communication range before it dropped, will cause safety to not be guaranteed. The following strategies handle this trade-off differently. Other approaches were implemented, but did not achieve sufficient success to be incorporated into the final report.

4.1.1 Benchmark Strategies

There are two benchmarks with which to compare any developed strategy. Firstly, we wish to consider a strategy which ignores safety and only cares about performance, which we will denote the performance benchmark. This benchmark will simply not account for the communication failure in any way, and will continue towards it goal assuming that non-communicating drones do not exist (ignoring previous trajectory information). This is not a viable strategy in real operations, as it is likely to cause collisions and is not making use of available information. However, it serves as a valuable benchmark to determine the optimal performance of the algorithm.

The second benchmark, which we will denote the safety benchmark, considers safety at the expense of performance. This strategy attempts to do the only provably safe action when communication fails, which is to remain still until communication returns. This is likely not viable in real situations as not only does it introduce large delays, but suddenly stopping may add a large amount of oscillations into the system. Collisions can still occur as the drones cannot stop in place instantly, and thus my collide while attempting to slow down and stay in their last communicated position. However, this approach should minimize collisions and serves as a simple benchmark.

4.1.2 Static BVC

The paper presenting the chosen algorithm [22] compares their approach to a buffered Voronoi cell (BVC) approach [40]. This approach calculates Voronoi cells for each drone, with a small buffer between them to ensure safety, and maintains the drones inside their own cell for a certain amount of time before recalculating the cells. This ensures safety at each time step if the current position of all the drones is known (assuming it is dynamically feasible to remain in the cell), or probable safety if the neighbouring drones are known. In [22], it was shown that this approach suffers from deadlock and is overly conservative compared to the proposed hyperplane on-demand collision avoidance. However, BVCs can be used to ensure safety while still allowing the drones to move forward.

This approach, which we will denote the BVC approach, will construct buffered Voronoi cells when communication fails using the last known position of neighbouring drones. Then, the drones can be constrained to remain within their cells until communication returns. This is essentially an improved version of the safety benchmark as safety is still guaranteed if the positions of all drones is known (and the drones can stay in their cells), but still allows the drones to move towards their goals. Even for smaller BVCs, the drone should have more space to decelerate than the safety benchmark that immediately tries to make the drone stationary, and thus this approach is expected to have less oscillation.

4.1.3 On Demand Using Previous Trajectories

The BVC approach in the previous section only makes use of the last known positions of the other drones. However, using the anticipated trajectories of the other drones we can proceed less conservatively. If there are no expected collisions on the horizon, and the desired trajectories of all the drones is known, and disturbances are assumed to be zero, each drone can continue along its original trajectory without pause. If a collision is detected along the horizon, the drones can calculate separating hyperplanes and avoid the collision as intended in the chosen algorithm, assuming that the other drone is following its original trajectory. However, since avoiding this collision will lead the drone (and its neighbours) to deviate from their originally intended trajectories, they may introduce a future collision without knowing it. To minimize this chance, a penalty term is added to the MPC cost function to minimize deviations from the originally shared trajectory, so the drones stay close to the trajectory originally shared with the other drones.

This approach does not prevent collisions but should minimize their probability, and when communication drops for short periods of time (less than the horizon), it may effectively reduce collisions while maintaining performance. However, if communication drops for longer than the horizon, it is unclear how to proceed. The other approaches could be used here, such as stopping completely (the safety benchmark), or continuing as if there are no other drones (the performance benchmark). However, to differentiate the approaches, when communication fails for longer than the horizon, the drone will assume the other drones are moving forward using the final velocity in their trajectory. Each drone will also be penalized for deviating from this trajectory, similar to before. However, since this approximation may deteriorate as time progresses and take the drones far from their intended goal, the speed of the drones will decrease until they are fully stopped. This should balance performance and safety less conservatively than before.

4.2 Types of Communication Failure

Communication can fail in a variety of ways. This includes delays and bandwidth limitations [17], limited range [27], blocked by obstacles [27], stochastic packet drops [32], or system-wide failure [31]. Although all real communication channels have delays and bandwidth limitations, and this is considered an important future work in the chosen paper, it is not the focus of this project. The chosen paper already implements limited range communication to reduce the computational overhead of considering every drone on every iteration, and this range is set to $3r_{\text{min}}$. There are no obstacles in our experiments, so this will not need to be considered. Thus, the main communication failures considered will be stochastic and pre-programmed system-wide failure.

The strategies considered in Section 4.1 generally only apply when all the drones lose communication at once. For example, for the safety benchmark and the BVC approach, it is unclear whether the drones must switch to their safe backup if only a portion of the other drones become disconnected. In general, it is unclear how the drones should act if they believe they are transmitting to the other drones, but are in fact not. For these reasons, we will only consider system-wide failure. The communication will drop system wide in two ways: stochastic, where on every iteration there is a chance c that communication fails for that iteration, and periodic, where there are periods of failed communication between half the horizon to 1.5 times the horizon in length that occur with a frequency such that the overall percentage of iterations where there is no communication is approximately c.

4.3 Testing Environment

The above strategies are tested in MATLAB, using the open-sourced code for [22], `https://github.com/carlosluis/online_dmpc`, as a foundation. Our code can be found at `https://github.com/Federico-PizarroBejarano/online_dmpc`.

The chosen trajectory generation algorithm has many tuning parameters and can be tested with a variable number of drones. To simplify the comparison of our approaches, we will vary only one parameter, the amount of communication failure c (described in Section 4.2). The other parameters are fixed, and detailed below:

- **Number of drones**: We will use 15 drones. This is a large number of drones for the testing space, which causes frequent collisions, while still being computationally inexpensive. Although this density of drones is impractical in real experiments due to the risk of collision, it served to better demonstrate the capabilities of our approaches.
- **Testing Space**: The drones have starting and ending locations within the box defined by $x \in [-1\,\text{m}, 1\,\text{m}]$, $y \in [-1\,\text{m}, 1\,\text{m}]$, and $z \in [0.5\,\text{m}, 2\,\text{m}]$, which is significantly smaller than the volume tested in the chosen algorithm. There are no obstacles.
- **MPC Horizon**: The horizon of the MPC was found to have a large effect on the behaviour of the algorithms, as longer horizons cause longer trajectories to be shared with neighbouring drones and collisions to be predicted earlier. However, it also results in an increase in computation. The horizon was set to 10 which was found to be a reasonable trade-off between performance and computational complexity.
- **Minimum Distance**: The minimum distance r_{min} the drones must stay away from one another (measured from their centres of mass) was chosen to be 30 cm.
- **Frequency**: The MPC runs at a frequency of 10Hz, which is sufficient as the algorithm sends position set-points rather than lower-level commands such as thrust which would require higher frequencies.

The chosen algorithm has many more tuning parameters, which were left at their default values.

4.4 Testing Methodology

Using the algorithm setup and environment in Section 4.3, experiments were conducted using the coordination strategies detailed in Section 4.1. Each experiment consisted of the drones starting at randomly generated locations, and traveling to randomly generated goal locations (where neither the start nor goal states violate collision constraints defined by the minimum distance). Once all the drones arrive at the goal state, they receive a new set of goal points. This is repeated 5 times, so each experiment is a series of 5 point-to-point transition tasks.

Five experiments were conducted per strategy per communication level, with levels $c \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. At $c = 1$, there is no communication and only the performance benchmark will converge. At $c = 0$, every approach simply becomes the underlying algorithm. Both of these are included as comparisons.

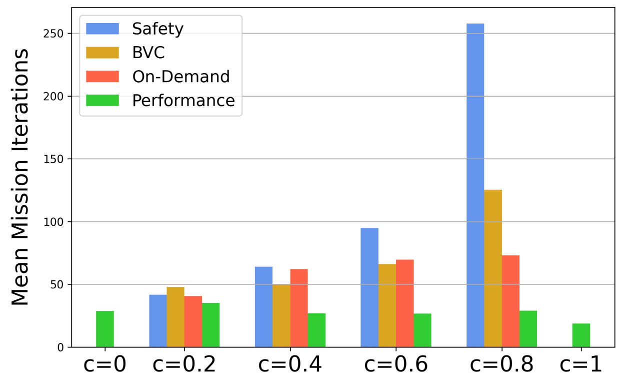
5 Results

5.1 Metrics

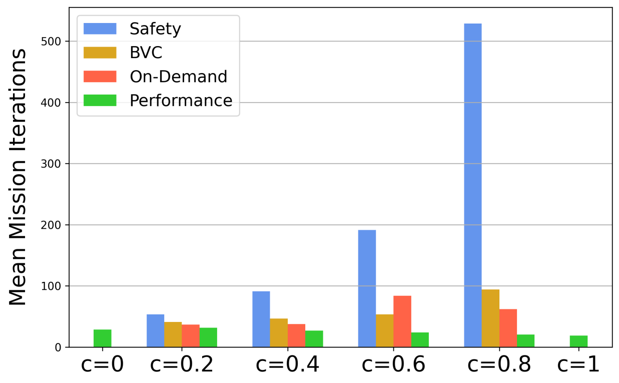
The experiments were conducted to compare the safety and performance of each strategy. As discussed in Section 4.1, there is a trade-off between guaranteeing safety and attempting to reach the goal points rapidly. Safety was measured as the number of collisions, where a collision is when two drones are closer than the minimum distance r_{min}. If two drones are colliding for three time steps, then it would be counted as three collisions. Performance was measured was the total number of MPC iterations required to complete the transition task, as the objective is to reach the goal points rapidly. Additionally, we wish to measure the computational complexity of each approach and the effect of reduced communications on it. To do this, we measure the average real time required to execute an iteration of the MPC loop per strategy.

Each experiment was executed multiple times. However, it was found that in almost all cases, the standard deviation of the results was exceedingly small, mainly with the exception of the safety benchmark when communication is very poor (i.e., $c = 0.8$). Due to this, the results are illustrated using bar graphs rather than box plots as it was found to be more clear.

5.2 Transition Time



(a) Periodic Dropout



(b) Stochastic Dropout

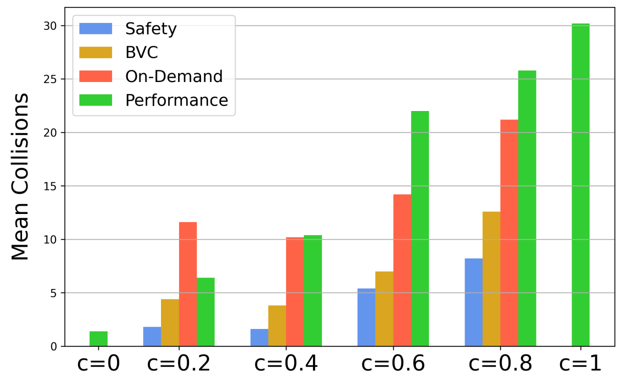
Figure 4: Average number of iterations necessary to successfully accomplish mission for each approach, for both periodic and stochastic communication dropout.

In Figure 4, we see the transition times per transition task for each proposed approach with each communication level c, for both periodic and stochastic communication dropout. The transition time is measured as the number of MPC iterations rather than the length of simulated time, as they are analogous.

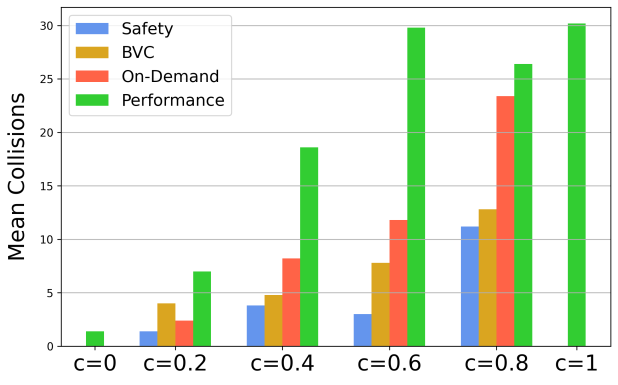
We immediately note that the two benchmark approaches, the safe benchmark (labeled 'Safety') and the performance benchmark (labeled 'Performance') follow expected patterns. Specifically, the safe approach increases the total mission time significantly as the communication deteriorates, while the performance approach actually decreases the mission time as it becomes less conservative as communication deteriorates. The other approaches generally increase the mission time as communication deteriorates as well, as they become increasingly conservative, with the BVC and the on-demand approaches being roughly equivalent.

We note that the mission times are similar between the periodic and stochastic communication dropout experiments, with the exception of the safety benchmark. In fact, the safety benchmark suffers greatly in the stochastic case, doubling the mission time as compared to the systemic case. This is due to the constant acceleration and deceleration caused by communication stochastically dropping and forcing the drones to attempt to remain still, an extremely inefficient and dangerous approach. The other approaches tend to fair similarly to slightly better when communication fails stochastically, as they have more recent information with which to plan.

5.3 Collisions



(a) Periodic Dropout



(b) Stochastic Dropout

Figure 5: Average number of collisions for each approach, for both periodic and stochastic communication dropout.

In Figure 5, we see the number of collisions per transition task for each proposed approach with each communication level c, for both periodic and stochastic communication dropout.

We immediately note that the safety benchmark has consistently the lowest number of collisions and the performance has one of the highest number of collisions, as expected. The BVC approach has levels close to the safety benchmark, while the on-demand approach tends to be closer to the performance benchmark.

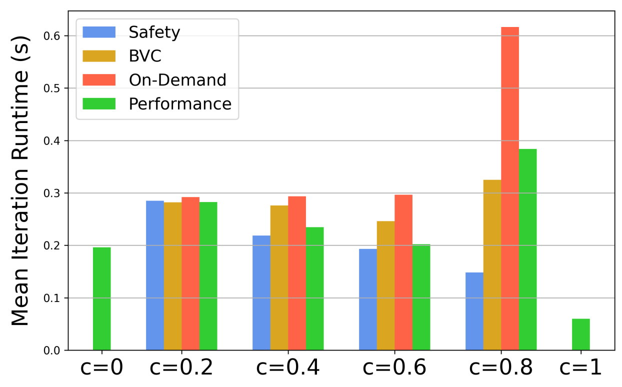
Surprisingly, the number of collisions is roughly similar between the periodic and stochastic communication dropout experiments. It was anticipated that the violations would be significantly higher in the periodic approach due to it planning with far less information. However, the instability generated by the stochastic approach forcing the MPC to switch often caused a large number of violations and instability generally.

5.4 Computation Time

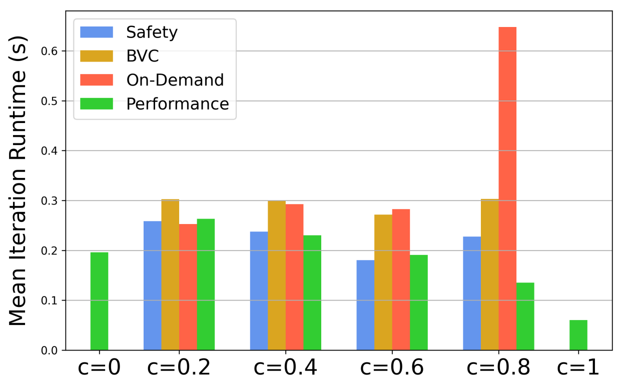
In Figure 6, we see the average runtime of one MPC iteration for each proposed approach with each communication level c, for both periodic and stochastic communication dropout. We note that the two benchmark approaches decrease slightly in runtime as communication deteriorates, as they are using simplified approaches when communication fails rather than the more complex collision avoidance of the other approaches. The BVC method maintains approximately the same computation time, which is expected as in the chosen solution it was found that the on-demand and BVC approaches have similar computational costs [22]. Surprisingly, the on-demand approach increases in computational cost significantly when communication is at the lowest level, $c = 0.8$, but is otherwise fairly constant. It is unclear why this is the case, but may be due to the on-demand approach's enforcement of maintaining the last shared trajectory moving the system away from the goal, and thus the MPC having greater difficulty balancing the objectives and constraints. Additionally, the runtime is similar in both the periodic and stochastic dropout experiments.

5.5 Discussion and Recommendations

When designing approaches, it became clear there was a trade-off between safety and performance, and the two benchmarks were created to highlight the best possible performance and one of the simplest yet safest approaches. In this regard, the benchmarks performed exactly as expected, consistently having the best



(a) Periodic Dropout



(b) Stochastic Dropout

Figure 6: Average runtime of one iteration for each approach, for both periodic and stochastic communication dropout.

performance and worse safety (in the case of the performance benchmark) and vice versa (in the case of the safety benchmark). The optimal approach would be one that closely approximates the safety of the safety benchmark and the performance of the performance benchmark.

From Figure 4 and Figure 5, we see that the BVC approach maintains similar performance top the on-demand approach in almost all experiments, and significantly reduced collisions compared to the on-demand approach. By enforcing the drones stay within their previously calculated BVC, safety is enforced but it still allows the drones to move toward their goals as well as not introducing the instability of starting and stopping of the safety benchmark. Additionally, it relies only on the last known position and thus can be applied to a more underlying algorithms, and it does not exhibit the spike in computation time when communication is low that the on-demand approach experiences.

Due to these reasons, if a trajectory generation algorithm that assumes perfect communication was to be implemented on a real multi-agent system, it is our recommendation that a simple yet effective backup to use when communication fails is the BVC method.

6 Conclusions and Future Work

6.1 Future Work

The exploration into different approaches to robustifying trajectory generation algorithms that assume perfect communication can be extended in several ways.

Firstly, the proposed approaches are only clearly defined when communication fails entirely for all drones at once. This is one possible type of communication failure, but another important case is when individual drones lose connection to some or all other drones. Currently, only the performance benchmark is clearly defined for this case, as it can simply ignore any non-communicating drone. The on-demand approach could be extended as well to account for this by performing on-demand collision avoidance with the last known trajectory of non-communicating drones. However, it is unclear how to ensure that the drones are following trajectories similar to their last communicated trajectories without overly constraining the trajectories of the drones. The BVC method could also be extended to enforce the drones to remain within their BVC compared to non-communicating drones, and use the normal on-demand approach within their BVC with communicating drones.

Perhaps a superior approach that was considered but not implemented would be a tube-based approach that assumes the non-communicating drones are within a increasing radius from their last known position. By avoiding this tube, and remaining within one's own tube, the drones could avoid one another while guaranteeing the unknown trajectories of the other drones are within a certain area. However, this may overly constrain the drones.

Additionally, comparisons to state-of-the-art approaches that allow for communication failure should be done, even though these approaches may be different than the baseline chosen approach.

Finally, in this work we have considered only outright communication failure, but when communication succeeds it is instantaneous. More realistic communication involves bandwidth limitations and delays, which should be modelled and accounted for, such as in [17]. When communication delays are small, it is likely this could be accounted simply. However, longer communication delays may require switching to a backup system, such as the ones presented in this project.

6.2 Conclusion

In this project, we explore the problem of generating safe and efficient trajectories for multi-agent systems when communication is unreliable. Firstly, we analyze the literature related to multi-agent systems. We motivate the necessity for decentralized algorithms for scalability and execution that is robust to communication failure. Next, we consider coordination strategies between decentralized agents, both explicit and implicit, and motivate the necessity of implicit coordination when communication fails. From this background, we choose a multi-agent trajectory generation algorithm [22] which has demonstrated state-of-the-art performance in real and simulation experiments. This algorithm served as the underlying algorithm with which we explore the effects of communication failure and approaches to implicit coordination.

Once the algorithm is chosen, several implicit coordination approaches are designed, including approaches meant to demonstrate the best possible performance and safety. An experiment with a high density of drones is created, with both stochastic and periodic system-wide communication failure. The approaches are run on this experiment, resulting in a overall measure of the best safety and performance, and approaches that attempt to balance the two. From these results, we conclude that the best approach for real-world execution is a buffered Voronoi cell approach where the drone is kept within its own Voronoi cell when communication fails, as it exhibits high safety and performance with minimal information and computation. Further research is necessary to test more realistic communication failures, as well as more implicit coordination approaches.

References

[1] Adajania, V. K., Zhou, S., Singh, A. K., and Schoellig, A. P., "AMSwarm: An Alternating Minimization Approach for Safe Motion Planning of Quadrotor Swarms in Cluttered Environments," *arXiv preprint arXiv:2303.04856*, 2023.

[2] Augugliaro, F., Schoellig, A. P., and D'Andrea, R., "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 1917–1922, 2012.

[3] Bautin, A., Simonin, O., and Charpillet, F., "MinPos : A Novel Frontier Allocation Algorithm for Multi-robot Exploration," in *International Conference on Intelligent Robotics and Applications*, 2012.

[4] Botelho, S. and Alami, R., "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1234–1239 vol.2, 1999.

[5] Capitan, J., Spaan, M. T., Merino, L., and Ollero, A., "Decentralized multi-robot cooperation with auctioned POMDPs," in *2012 IEEE International Conference on Robotics and Automation*, pages 3323–3328, 2012.

[6] Cardona, G. A., Yanguas-Rojas, D., Arevalo-Castiblanco, M. F., and Mojica-Nava, E., "Ant-Based Multi-Robot Exploration in Non-Convex Space without Global-Connectivity Constraints," in *2019 18th European Control Conference (ECC)*, pages 2065–2070, 2019.

[7] Das, B., Couceiro, M. S., and Vargas, P. A., "MRoCS: A new multi-robot communication system based on passive action recognition," *Robotics and autonomous systems*, 82:46–60, 2016.

[8] de Hoog, J., Cameron, S., and Visser, A., "Dynamic team hierarchies in communication-limited multi-robot exploration," in *2010 IEEE Safety Security and Rescue Robotics*, pages 1–7, 2010.

[9] Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Stewart, B., "Distributed Multirobot Exploration and Mapping," *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

[10] Gerkey, B. P. and Matarić, M. J., "Sold!: auction methods for multirobot coordination," *IEEE Trans. Robotics Autom.*, 18:758–768, 2002.

[11] Hollinger, G. and Singh, S., "Multi-robot coordination with periodic connectivity," in *2010 IEEE International Conference on Robotics and Automation*, pages 4457–4462, 2010.

[12] Hollinger, G. A., Singh, S., Djugash, J., and Kehagias, A., "Efficient Multi-robot Search for a Moving Target," *The International Journal of Robotics Research*, 28:201 – 219, 2009.

[13] Hönig, W., Preiss, J. A., Kumar, T. K. S., Sukhatme, G. S., and Ayanian, N., "Trajectory Planning for Quadrotor Swarms," *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[14] Joy, K., "Bernstein polynomials," 2000.

[15] Karabag, M. O., Neary, C., and Topcu, U., "Planning Not to Talk: Multiagent Systems that are Robust to Communication Loss," *arXiv preprint arXiv:2201.06619*, 2022.

[16] Kemna, S., Rogers, J. G., Nieto-Granda, C., Young, S., and Sukhatme, G. S., "Multi-robot coordination through dynamic Voronoi partitioning for informative adaptive sampling in communication-constrained environments," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2124–2130, 2017.

[17] Kondo, K., Figueroa, R., Rached, J., Tordesillas, J., Lusk, P. C., and How, J. P., "Robust MADER: Decentralized Multiagent Trajectory Planner Robust to Communication Delay in Dynamic Environments," *arXiv preprint arXiv:2303.06222*, 2023.

[18] Liu, C. and Kroll, A., "A Centralized Multi-Robot Task Allocation for Industrial Plant Inspection by Using A* and Genetic Algorithms," in *International Conference on Artificial Intelligence and Soft Computing*, 2012.

[19] Liu, T.-M. and Lyons, D. M., "Leveraging area bounds information for autonomous decentralized multi-robot exploration," *Robotics Auton. Syst.*, 74:66–78, 2015.

[20] Liu, Y. and Nejat, G., "Robotic Urban Search and Rescue: A Survey from the Control Perspective," *Journal of Intelligent and Robotic Systems*, 72(2):147–165, 2013.

[21] Luis, C. E. and Schoellig, A. P., "Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control," *IEEE Robotics and Automation Letters*, 4(2):375–382, 2019.

[22] Luis, C. E., Vukosavljev, M., and Schoellig, A. P., "Online Trajectory Generation with Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020, arXiv:1909.05150 [cs].

[23] Luna, R. and Bekris, K. E., "Efficient and complete centralized multi-robot path planning," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275, 2011.

[24] Matignon, L., Jeanpierre, L., and Mouaddib, A.-I., "Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Processes," *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):2017–2023, 2021.

[25] Mohamed, K., El Shenawy, A., and Harb, H., "A Hybrid Decentralized Coordinated Approach for Multi-Robot Exploration Task," *The Computer Journal*, 62(9):1284–1300, 2018.

[26] Mottaghi, R. and Vaughan, R. T., "An integrated particle filter and potential field method applied to cooperative multi-robot target tracking," *Autonomous Robots*, 23:19–35, 2007.

[27] Murphy, R., Casper, J., Hyams, J., Micire, M., and Minten, B., "Mobility and sensing demands in USAR," in *26th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, page 138–142 vol.1, 2000.

[28] Nestmeyer, T., Franchi, A., Bülthoff, H. H., and Giordano, P. R., "Decentralized Multi-target Exploration and Connectivity Maintenance with a Multi-robot System," *ArXiv*, abs/1505.05441, 2015.

[29] Omidshafiei, S., Agha-mohammadi, A., Amato, C., and How, J. P., "Decentralized Control of Partially Observable Markov Decision Processes using Belief Space Macro-actions," *CoRR*, abs/1502.06030, 2015.

[30] Schoellig, A., Augugliaro, F., and D'Andrea, R., "A Platform for Dance Performances with Multiple Quadrocopters," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010: 18 - 22 Oct. 2010, Taipei, Taiwan*, IEEE, 2010, accepted: 2017-06-14T16:15:54Z.

[31] Tan, A. H., Bejarano, F. P., Zhu, Y., Ren, R., and Nejat, G., "Deep Reinforcement Learning for Decentralized Multi-Robot Exploration With Macro Actions," *IEEE Robotics and Automation Letters*, 8(1):272–279, 2023.

[32] Turpin, M., Michael, N., and Kumar, V. R., "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, 33:143–156, 2012.

[33] Viseras, A., Xu, Z., and Merino, L., "Distributed Multi-Robot Cooperation for Information Gathering Under Communication Constraints," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1267–1272, 2018.

[34] Wei, C., Hindriks, K. V., and Jonker, C. M., "Dynamic task allocation for multi-robot search and retrieval tasks," *Applied Intelligence*, 45(2), 2016.

[35] Xu, Z., Fitch, R., and Sukkarieh, S., "Decentralised coordination of mobile robots for target tracking with learnt utility models," in *2013 IEEE International Conference on Robotics and Automation*, pages 2014–2020, 2013.

[36] Yamauchi, B., "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998.

[37] Yan, Z., Jouandeau, N., and Chérif, A. A., "Multi-robot Decentralized Exploration using a Trade-based Approach," in *International Conference on Informatics in Control, Automation and Robotics*, 2011.

[38] Yan, Z., Jouandeau, N., and Cherif, A. A., "A Survey and Analysis of Multi-Robot Coordination," *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

[39] Zhang, H., Chen, J., Fang, H., and Dou, L., "A role-based POMDPs approach for decentralized implicit cooperation of multiple agents," in *2017 13th IEEE International Conference on Control and Automation (ICCA)*, pages 496–501, 2017.

[40] Zhou, D., Wang, Z., Bandyopadhyay, S., and Schwager, M., "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells," *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.