EECE 5550 Final Project Report

Gaurav Kothamachu Harish
Khoury College of Computer Science
Northeastern University
Boston, MA
kothamachuharish.g@northeastern.edu

Rachel Lim
College of Engineering
Northeastern University
Boston, MA
lim.rac@northeastern.edu

Savannah Macero
College of Engineering
Northeastern University
Boston, MA
macero.sa@northeastern.edu

Aaron Pan
College of Engineering
Northeastern University
Boston, MA
pan.aa@northeastern.edu

***Abstract*—We aim to evaluate a modular Bayesian optimization framework for efficiently generating time-optimal trajectories for cooperative multi-agent systems. The proposed algorithm uses a Bayesian optimization model containing multiple Gaussian process surrogate models, which model the agent's dynamic feasibility and collision avoidance constraints. Our implementation adapts an existing multi-fidelity approach of Bayesian optimization on a single drone, to a multi-drone single-fidelity case. We also simplify the problem to 2 drones with a reduced number of polytopes and waypoints to save on compute resources and evaluate results. We show that the modular Bayesian optimization framework does indeed generate time-optimal trajectories for 2 drones that avoid collisions and obstacles. However, the algorithms' quick generation of time-optimal trajectories are offset by the expensive optimization time needed to generate the initial dataset for the Bayesian optimization framework. More specifically the initial datasets took over 4 hours to generate for each drone, while the Bayesian optimization algorithm took a couple of minutes to execute. Thus, we believe that the Bayesian optimization algorithm is indeed a faster way to generate trajectories quickly, but at the expense of initial dataset generation time. A link to our Github repository can be found here: github.com/rachel-lim/multi-agent-bayes-opt**

I. Introduction & Motivation

Fast navigation of autonomous vehicles, specifically that of unmanned aerial vehicles (UAVs), is a popular area of research due to its breadth of application across topics and the intrinsic complexity of fast and agile dynamical control. For useful real-world applications, non-trivial external factors must be considered, such as dynamic or cluttered workspaces or the addition of cooperative agents. The latter introduces a particularly difficult set of challenges for roboticists, one of which can be referred to as the curse of dimensionality. The number of variables increases proportionally to the number of agents, and the volume of the data space increases exponentially with dimension [1]. Opposed to the single-agent problem, a multi-agent problem needs to be optimized for each agent trajectory in the system.

UAVs, or drones, are a robust robotic platform choice for implementing cooperative multi-agent systems due to their accessibility, agility, and modularity. They have been employed with cooperative motion planning for a diverse array of scenarios ranging from team-based surveillance [2] to drone light show performances [3].

The paper we chose to focus our project on presents a modular Bayesian optimization architecture to generate time-optimal trajectories for a cooperative multi-agent drone system that must simultaneously reach specific waypoints [1]. This waypoint-constrained framework could be applied to the planning for multi-agent systems in which agents must attain formations quickly and simultaneously, such as data sharing and collecting for search and rescue operations [4], or in team-based defense against a shared adversary [5]. A diagram of the proposed algorithm with two drones can be seen in Fig.1.

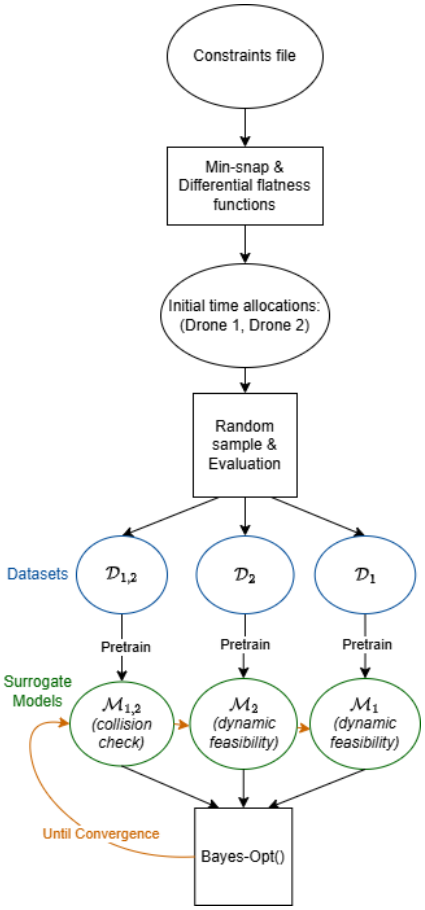

Fig. 1. Flowchart of initial dataset generation to running the Bayes-Opt() function, also seen in Fig 2. This flow chart represents the initial dataset ($\mathcal{D}$) and surrogate model ($\mathcal{M}$) generation which are then used in the Bayes-Opt() function.

We collectively found interest in how path planning is approached for multi-agent systems, and were curious about how this paper utilized Bayesian optimization to address the

curse of dimensionality. The authors adapted the training procedure of the surrogate model and presented novel acquisition function formulations such that the algorithm trains individual vehicles for dynamic feasibility and pairwise vehicle surrogate models for collision [1].

In completion of this project, we hoped to recreate the code to run the trajectory generation to validate their method in generating time-optimal solutions under the provided constraints.

II. Problem Statement

The goal of the paper is to produce time-optimal trajectories for multiple drones, subject to collision avoidance and simultaneously reaching specific waypoints. Each drone is provided a start point, end point, and set of waypoints, and must avoid obstacles in the environment. Additionally, the set of drones must reach each set of waypoints simultaneously, and avoid colliding with each other.

To reduce the computation load of multi-agent optimization, we utilize Bayesian optimization to do feasibility sampling of the search space and build surrogate models representing the dynamics and collision constraints. The rest of this section is a summary of the methods described in the paper and cited works:

For a set of trajectories with N_v drones and N_f waypoints we can describe the problem with:

$$\begin{aligned}
&\min_{\mathbf{p},\mathbf{t},\mathbf{T}} \max_{i=1...N_v} T_i \\
&\text{s.t.} \\
&\quad p_i(0) = p_i^{\sim start} \\
&\quad p_i(T_i) = p_i^{\sim end} && i = 1,...N_v \\
&\quad p_i(t_k^{form}) = p_{i,k}^{\sim form} && i = 1,...N_v, k = 1,...N_f \\
&\quad t_k^{form} \leq t_{k+1}^{form} && k = 1,...N_f - 1 \\
&\quad t_{N_f}^{form} \leq T_i \\
&\quad p_i \in \mathcal{P}_i \\
&\quad p_i \in \mathcal{F}_{T_i} && i = 1,...N_v \\
&\quad (p_i, p_j) \in \mathcal{F}_{T_i,T_j} && i,j = 1,...N_v, j > i
\end{aligned} \tag{1}$$

Where p represents a continuous function with $p(t) = [p_r(t)^T, p_\psi(T)]^T$ is the position and yaw at time t.

The times of formations is given by $\mathbf{t} = [t_1^{form}, ...t_{N_f}^{form}]$, set of trajectories is $\mathbf{p} = \{p_i | i = 1, ...N_v\}$, and $\mathbf{T} = [T_1, ...T_{N_v}]$.

For a vehicle i, $p_i^{\sim start}$ is the start position, $p_i^{\sim end}$ is the end position, and the waypoints are given by $p_{i,k}^{\sim form}$ for $k = 1, ...N_f$.

The set of valid trajectories is given by the intersection of three spaces, $\mathcal{P}_{\mathcal{T}_i}$ is the set of trajectories that satisfy dynamic feasibility constraints for $t = [0, T_i]$, $\mathcal{F}_{T_i}$ are the trajectories that avoid obstacles in the environment, and $\mathcal{F}_{T_i,T_j}$ are the trajectory pairs that don't collide.

A. Dynamic feasibility

To check dynamic feasibility, we use the differential flatness method on idealized quadrotor dynamics. A differentially flat system is one where we can completely describe the outputs through the state and controls. Using the flatness transform allows us to find the control inputs corresponding to a trajectory, and evaluate those inputs (i.e. motor speeds) for feasibility [6]. The original paper also uses a multicopter dynamics simulator provided in [7] which we will discuss more in the next section.

B. Obstacle Avoidance

We can describe the free space in an environment using polytopes, and formulate the obstacle avoidance constraints using linear inequalities for the polytope constraints.

For a polygon Q with d faces, we can describe the polygon as $Q = \{p_0, n_0, ...p_{d-1}, n_{d-1}\}$ where p_j, n_j are the adjacent vertex and normal vector of the jth face. We can ensure a trajectory segment stays within the polytope with:

$$Ap(t) \leq b \tag{2}$$

$$A = [n_0, ...nd - 1]^T \tag{3}$$

$$b = [n_0^T p_0, ...n_{d-1}^T p_{d-1}]^T \tag{4}$$

We can ensure obstacle avoidance through a trajectory by breaking up the overall trajectory into a segment per polytope. That is, for N_p polytopes we can describe the time allocations as $\mathbf{x} = [x_1, ...x_{N_p}]$, and the polytope constraints become $A_i p(t) \leq b_i \forall t \in [x_{i-1}, x_i], i = 1, ...N_p$ where $x_0 = 0$.

C. Waypoint Formations

We ensure the multiple vehicles reach their waypoints simultaneously by segmenting the trajectory by the number of waypoints, and ensuring these time segments are equal across the vehicles. For N_f waypoints and N_v vehicles, we have $\mathbf{x} = [x_1, ...x_{N_f}]$ and $p_k(\sum_{i=1}^{j} x_i) = p_k^{\sim form} \forall j = 1, ...N_f, k = 1, ...N_v$.

We combine the polytope and waypoint constraints by breaking the trajectory into $m = N_p + N_f$ segments, turning the time allocations into $\mathbf{x} = [x_1, ...x_{e_1}, ...x_m]$ where e_k is the index of the trajectory segment that ends at waypoint k.

D. Minimum-Snap Method

We utilize the minimum-snap method, which writes feasibility as the objective rather than a constraint to reduce complexity [6]. Since feasible motor speeds are more likely at smaller control moments, we want lower angular accelerations, which can be achieved by minimizing snap (fourth order derivative of position) and the second order derivative of yaw.

The objective function and constraints for each drone can be written as:

$$\min_p \sigma(p, T) = \int_0^T \mu_r \left\| \frac{d^4 p_r}{dt^4} \right\|^2 + \mu_\psi \left(\frac{d^2 p_\psi}{pt^2} \right)^2 dt \tag{5}$$

s.t.

$$p(0) = p^{\sim start} \tag{6}$$

$$p(T) = p^{\sim end} \tag{7}$$

$$T = \sum_{i=1}^{m} x_i \tag{8}$$

$$A_i p(t) \leq b_i \qquad \forall t \in \left[\sum_{j=1}^{i-1} x_j, \sum_{j=1}^{i} x_j\right], i = 1, ...m \tag{9}$$

$$p\left(\sum_{=1}^{e_k} x_i\right) = p_k^{\sim form} \qquad e_k = 1, ...N_f \tag{10}$$

Where (3) and (4) are the start and endpoint constraints, (6) are the polytope constraints described in (II-B), and (7) are the waypoint constraints described in (II-C).

To minimize the time allocations, we start with an initial, large guess for total trajectory time and adjust the trajectory segment time allocations to find the minimum snap trajectory. Then we minimize time by scaling down the total time while ensuring the new trajectory is feasible using the procedure described in (II-B):

$$\min_x \sigma(\mathcal{X}(x, \tilde{F}), T) \tag{11}$$

$$\text{s.t. } T = \sum_{i=1}^{m} x_i$$

$$\min_\eta T \tag{12}$$

$$\text{s.t. } T = \sum_{i=1}^{m} \eta x_i$$

$$\mathcal{X}(\eta x, \tilde{F}) \in \mathcal{P}_T$$

where $p = \mathcal{X}(x, \tilde{F})$ is the solution from (5) and $\tilde{F}$ represents the set of constraints.

To generate our initial dataset, we use the min-snap method for each drone individually. However, after the initial segment time allocations for each drone are calculated, we scale them based on the allocations calculated for the other drones to ensure all drones reach their waypoints simultaneously. Additionally, we choose the largest scaling factor that is valid for all the drones to ensure the same total trajectory time.

E. Collision Checking

The simplest collision checking method simply enforces a minimum distance constraint between discrete points on each trajectory. While this would create a nonlinear optimization problem if used as a constraint, we avoid this issue since we only use it to generate a dataset for training our surrogate models.

F. Bayesian Optimization

Bayesian optimization is a method for optimizing black-box functions, where the objective or constraint functions are unknown. We use function evaluations to form a posterior distribution and construct an acquisition function, which determines the next sampling points.

Gaussian process classification (GPC) is a method of forecasting a function's output given a set of input data by modeling the probability distribution over possible functions. Given data points $X = \{x_1, ...x_n\}$, evaluations $y = \{y_1, ...y_n\}$, and latent variables $f = \{f_1, ...f_n\}$ we can predict the probability of test point $x*$ based on the latent variables with:

$$P(y*|x*, X, y) = \int P(y*|f*)P(f*|x*, X, y)df* \tag{13}$$

The acquisition function, $\alpha(x|\mathcal{D})$, gives the value of a point x given the dataset $\mathcal{D}$. We choose the next evaluation point with:

$$x_{next} = \arg\max_x \alpha(x|\mathcal{D}) \tag{14}$$

To ensure both the individual drone dynamics and collision avoidance are learned, we build models for both individual and all possible pairs of drones. For an individual drone model $\mathcal{M}_i$, we train it on $\mathcal{D}_i$, which includes a set of time allocations and their evaluations based on dynamic feasibility and obstacle avoidance. For drone pairs, we build $\mathcal{M}_{i,j}$ with $\mathcal{D}_{i,j}$ which evaluates their collision avoidance.

To balance exploration and exploitation, we define the acquisition function as follows:

$$\alpha(x) = \begin{cases} \alpha_{exploit}(x) & \text{if } \exists x \in \mathcal{X} \text{s.t.} \alpha_{exploit}(x) > 0 \\ \alpha_{explore}(x) & \text{otherwise} \end{cases} \tag{15}$$

$$\alpha_{explore}(x) = -\sum_{i=1}^{N_v} \frac{|\mu(x_i)|}{\sigma(x_i)} - \sum_{i=1}^{N_v - 1} \frac{|\mu(x_i, x_j)|}{\sigma(x_i, x_j)} \tag{16}$$

$$\alpha_{exploit}(x) = \begin{cases} \alpha_{EI}(x)\tilde{P}(y = 1|x) & \text{if } \tilde{P}_i \geq h, \tilde{P}_{i,j} \geq h \forall i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\mathcal{X}$ is a randomly generated set of sample trajectories. The expected improvement is given by:

$$\alpha_{EI} = \max_{i=1,...N_v} \sum_j \bar{x}_{ij} - \max_{i=1,...N_v} \sum_j x_{ij} \tag{18}$$

where x_{ij} is the ith vehicle's time allocation to the jth segment, and $\bar{x}_{ij}$ is the current best solution.

The probability a set x of time allocations is feasible is given by:

$$\tilde{P}(y = 1|x) = \left(\prod_{i=1}^{N_v} \tilde{P}_i\right)\left(\prod_{i=1}^{N_v - 1} \prod_{j=i+1}^{N_v} \tilde{P}_{i,j}\right) \tag{19}$$

$$\tilde{P}_i = \tilde{P}(y_i = 1|x_i) \tag{20}$$

$$= P(\mu(x_i) - \beta\sigma(x_i) \geq 0|x_i) \tag{21}$$

$$\tilde{P}_{i,j} = \tilde{P}(y_{i,j} = 1|x_i, x_j) \tag{22}$$

$$= P(\mu(x_i, x_j) - \beta\sigma(x_i, x_j) \geq 0|x_i, x_j) \tag{23}$$

where $\mu(x_i)$ and $\sigma(x_i)$ are the mean and standard deviation of the model's posterior distribution, $P(f|x_i, \mathcal{D}_i)$ and similarly $\mu(x_i, x_j)$ and $\sigma(x_i, x_j)$ are the mean and standard deviation of the two drone model's posterior distribution $P(f|x_i, x_j, \mathcal{D}_{i,j})$.

While sampling data points and evaluating the acquisition function on said points are traditionally done sequentially, we combine them for higher efficiency. We generate a set of random time allocations for the trajectory segments, and use that to rescale the sampled candidate data points from each drone. We check the probability, $P(y_i = 1|x_i, \mathcal{D}_i)$ of the feasibility of each data point and discard those with low probabilities. With the remaining feasible points, we check the collision-avoidance probabilities, $P(y_{i,j} = 1|x_i, x_j, \mathcal{D}_{i,j})$ and again discard those with low probabilities (i.e. those likely to collide).

We can use the remaining points to generate the next evaluation points following our acquisition function strategy (15). The evaluation points are added into their corresponding datasets, and the models are updated accordingly.

The algorithm that we followed can be found in Fig. 2.

Algorithm 1: Modular Bayesian optimization with simultaneous acquisition function evaluation

Input: Surrogate model $\mathcal{M}_1, \cdots, \mathcal{M}_{N_v}$, $\mathcal{M}_{1,2}, \cdots, \mathcal{M}_{N_v-1,N_v}$, acquisition function α, size of candidate data points N_s

subfunction `SampleTraj(`i, $\mathcal{X}_F$`)`**:**
- $\mathcal{X} = \emptyset$
- **while** $|\mathcal{X}| < N_1$ **do**
  - $\mathcal{X}_t \leftarrow$ Randomly sampled N_s points
  - Rescale $\mathcal{X}_t$ with $\mathcal{X}_F$
  - Remove $\mathbf{x} \in \mathcal{X}_t$ s.t. $\tilde{P}_i(y_i = 1|\mathbf{x}, \mathcal{M}_i) < C_1$
  - $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{X}_t$
- **return** $\mathcal{X}$

function `Modular Bayes-Opt()`**:**
- **repeat**
  - $\mathcal{X}_F \leftarrow$ Randomly sampled N_s points
  - $\mathcal{X} = \mathcal{X}_F$
  - **while** $|\mathcal{X}| < N_2$ **do**
    - $\mathcal{X}_t \leftarrow$ SampleTraj(1, $\mathcal{X}_F$)
    - **for** $i = 2, \cdots, N_v$ **do**
      - $\mathcal{X}_t \leftarrow \mathcal{X}_t \times$SampleTraj($i$, $\mathcal{X}_F$)
      - **for** $j = 1, \cdots, i-1$ **do**
        - Remove $\mathbf{x} \in \mathcal{X}_t$ s.t. $\tilde{P}_{j,i}(y_{j,i} = 1|\mathbf{x}, \mathcal{M}_{j,i}) < C_2$
    - $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{X}_t$
  - $\mathbf{x}, l \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}, l \in \{l^1, \cdots, l^L\}} \alpha(\mathbf{x}, l|\mathcal{D})$
  - Evaluate $\mathbf{x}$ in l-th fidelity experiment
  - Update dataset $\mathcal{D}_{l,i}, \mathcal{D}_{l,i,j}$
  - Update the surrogate model $\mathcal{M}_{l,1}, \cdots, \mathcal{M}_{l,N_v}, \mathcal{M}_{l,1,2}, \cdots, \mathcal{M}_{l,N_v-1,N_v}$
- **until** *convergence of solution*
- $p_i^* \leftarrow \chi(\mathbf{x}_i^*, \tilde{\mathcal{F}}_i)\ \forall i = 1, \cdots, N_v$
- $T^* \leftarrow \max_{i=1,\cdots,N_v} \sum_{j=1} x_{ij}^*$
- **Output:** T^*, $[p_1^*, \cdots, p_{N_v}^*]$

Fig. 2. Main Bayes-Opt function from [1]

III. Proposed Solution

We start with the codebase linked from [8], which performs Bayesian optimization for a single drone with multi-fidelity simulations. The paper [8] uses experimental data from multiple fidelities to generate optimal trajectories for a single drone. They merge low-fidelity feasibility data from differential flatness tests, medium fidelity from a multirotor simulator, and high fidelity from real-life simulations. We used their framework with the following changes:

- Enforce waypoints: Modified initial trajectory generation functions and min-snap functions to guarantee drones reached waypoints simultaneously
- Multiple drones: Added a collision checking function, added dataset generation and model functions for drone pairs, and modified acquisition function to consider both single and pairwise drone datasets
- Single fidelity: Modified surrogate models and acquisition functions to use data of a single fidelity level
- Bayesian optimization: Added sample trajectory generation functions that consider probabilities of success of both individual drone dynamics and pairwise collision avoidance, ensured consistent time scaling between vehicles for waypoint formation
- Simplified environment: Created a simplified environment with fewer drones and waypoints to reduce computation time

A. Environment

While we initially set out to replicate the results from the paper exactly to compare our implementations, we had to simplify the geometry of the robotic workspace and reduced the number of drones to achieve realistic computation times. Trajectory 1 in [1] had 4 drones passing through 8 polytopes and 4 sets of formations; we created an environment with 2 drones, 3 polytopes, and 3 sets of formations.

We did this to reduce the number of models to train – under the mBO framework, a cooperative multi-agent team requires $N_v + \binom{N_v}{2}$ surrogate models, i.e. the sum of the number of vehicles and number of pairwise combinations.

Our simplified environment after applying convex polygonal decomposition is shown in Fig. 3. To generate the initial multi-agent trajectories, we define the environment's boundaries as a single non-convex simple polygon [9] and use the existing machinery in [8]'s code to decompose the free space into a set of four convex polygons.

B. Waypoints

We model the points on the polygon boundary planes as "soft points," i.e. points with dynamic values that change as the trajectories are optimized, and waypoints as "hard points," i.e. values that the user inputs for a given problem statement. To handle this we add a new entry in the environment which allows us to specify the types of points, and use that in the min-snap optimization (10). We use these waypoints to break

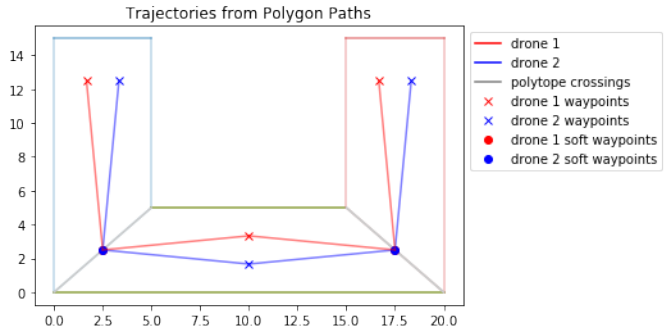

Fig. 3. Simplified environment for two drones, with the start, end, and waypoints for each drone marked (drone 1 in red, drone 2 in blue). The polytopes represent the free space in the environment, where the colored lines represent walls and the gray lines represent the plane between polytopes. Lines represent the resulting segmentation of trajectories given polytopes and waypoint constraints.

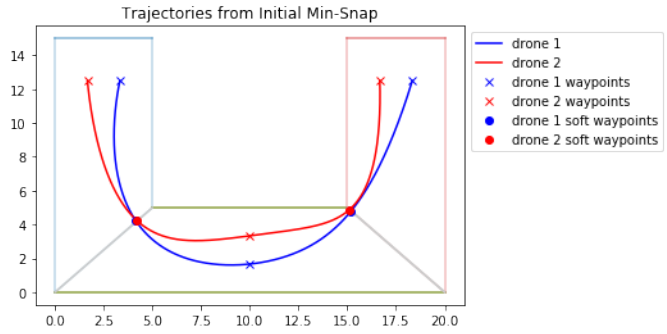

Fig. 4. Trajectories generated for each drone using min-snap optimization. While each trajectory is dynamically feasible and avoids obstacles, there is no collision checking and these paths do crash.

trajectory segments within a polytope into multiple segments to enforce the simultaneous formation constraint.

C. Multi-Drone

We added a collision checking function, used for the initial dataset generation and sample evaluation, that converts trajectory segment times into trajectory points using the min-snap optimization with differential flatness dynamic feasibility checks. We consider drones pairs that are within a specified Euclidean distance to have crashed.

We follow the method described in [1] to generate initial trajectories for the individual drones using the min-snap method. While we do ensure all the trajectory segments between waypoints sum to the same total time between the drones, and use the same scaling factor for the multiple trajectories, we do not check collision as can be seen in Fig. 4.

To learn collision avoidance in addition to dynamic feasibility and obstacle avoidance, we create datasets and models for both the individual and pairs of drones. In our environment, we have datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, consisting of individual drone trajectory time segments and the evaluation of dynamic feasibility and obstacle avoidance, used to train models $\mathcal{M}_1$ and $\mathcal{M}_2$. We additionally have $\mathcal{D}_{1,2}$, a set of trajectory times for both drones and their evaluations of collisions, used to train $\mathcal{M}_{1,2}$, which can be seen in Fig. 1. We use the same GPC method from [1] for all the models.

D. Bayesian Optimization

Our major change for the Bayesian optimization function is in how the sample set $\mathcal{X}$ is generated. Following Algorithm 1 (Fig. 2), we start with random sampling of time scalings, and use eq. (20) and (22) to evaluate the probability of success using the models, keeping a sample set more likely to be feasible.

To ensure the waypoints are visited synchronously by the two drones, we rescale the random sampled points for each drone χ_t by an initial sample, χ_F so that waypoints are visited synchronously.

We adjust the cutoff probabilities C_1 and C_2 during runtime if there are no points in the batch that reach the current cutoff, and increase it back to the original values as the model improves.

We modified the acquisition function following (15) to consider both the individual and pairwise models. Once we select a sample from the sample set $\mathcal{X}$, we evaluate those trajectory times for feasibility (both individual and collision avoidance) and add this information to the relevant datasets, and update all the models.

E. Other Changes

We used the differential flatness for all dynamic feasibility evaluations, while we did run a few tests using the simulator they were prohibitively slow. We also consider min-snap trajectories that fail to optimize and return the initial connect-the-dots trajectory to be infeasible.

We set both C_1 and C_2 to 0.7 (down from 0.8 in the paper), and decrease it by 0.01 for each iteration where no feasible points are found. We left the other acquisition function parameters, $h_1 = 0.001$ and $\beta_1 = 3.0$ the same as the paper.

We used the minimum Euclidean distance between drones for collision checking where less than 0.2m was considered a crash.

We ran each models for 500 epochs, and the overall Bayesian optimization for 200 iterations.

IV. Results

All results below are run in the two drone; 3 polytope; and start, end, and 1 set of waypoint constraints shown in Fig.3.

We present our results as follows: The first is generating initial datasets for each drone and discussing the associated compute time. Then we use the Bayesian optimization to train and optimize the surrogate models for dynamic feasibility checks and collision constraints.

A. Initial Trajectory Generation

The min-snap trajectories that we sampled for the initial dataset are shown in Fig. 4. Since these two trajectories are generated separately, they are time-optimal, avoid obstacles,

and reach the waypoints synchronously, but we can see them collide at two points.

Generating the initial dataset from that trajectory was an extremely slow process due to the number of samples tested and all the feasibility evaluations, even when using the differential flatness method. We experimented with generating 100, 1000, and 2000 data points, and found that 100 points wasn't enough to create an optimal model (model had a high loss even after many iterations of training and produced many infeasible trajectories). However, 2000 points did not yield substantial improvements when compared to 1000, so we used 1000 points for faster dataset generation. However, even with 1000 points it still took about 4 hours to generate the datasets for each drone.

We noticed when generating the samples for individual drones that the vast majority of samples passed feasibility checks, but we needed to generate a final dataset with an equal number of passing and failing checks. One possibility to speed up this dataset generation process would be to do a pseudo-random generation by training our model, or just calculating the Gaussian distribution, of passing and failing trajectories and primarily sampling from that to find more failing trajectories. However, we weren't sure how much worse of a model that would create.

The dataset generation time raises a drawback of this method. The Bayesian optimization algorithm runs on the order of minutes to converge to the solution, but this is offset by the massive training time used to generate initial datasets. In the paper's original multi-fidelity method, that may be less of a drawback since both the simulation and real-world experiments are significantly more time expensive than this method, but it still limits the applications to scenarios where an extremely time-optimal solution is key but large pre-training times are reasonable.

In addition, the surrogate models cannot generalize to different obstacles or waypoints, since the time scalings learned are specific to the details of an environment (although they could be reused if an additional drone is added into the formation).

B. Trajectory Generation from Bayesian Optimization Function

We trained both the individual drone and the pair models for 500 epochs. The individual drones are able to reach a loss of 0.001 (starting at 0.004), and the pair goes from 0.004 to 0.002.

The acquisition model almost always generates trajectories that meet individual drone feasibility checks, but struggled significantly more in collision avoidance. It is possible we need more samples for the pairwise drone models, as that dataset has 8 dimensions instead of 4. We may also have to tune the cutoff probabilities separately, or set a lower boundary on how far the cutoffs can drop even if no valid samples are found.

We tested initial datasets that included only pairs of time arrays that were scaled to ensure waypoint formation, as well as non-synced time arrays, but both produced similar results during the optimization.

The optimization loop also typically runs quite slowly, taking a couple of hours per hundred iterations. We were not able to run it convergence, but did see improvements in both the total trajectory time and collision avoidance although due to the random sampling, the results did sometimes get worse. The plots of trajectories evaluated throughout the optimization process are shown in Fig. 5.

Most of the time taken during the optimization was for generating a very large sample set to find enough points that met the feasibility probability for all the models. Creating larger initial datasets, training the models for more epochs, or lowering the probability threshold may help with runtime.

V. Conclusion

We believe we were able to recreate the optimization algorithm proposed in [1], and were able to create dynamically feasible, obstacle-avoiding, and collision-avoiding trajectories and see the time improve over iterations. However, this comes at a very high computation time, both for generating the initial datasets needed to train the surrogate models and for running the optimization iterations. We additionally only did the fastest feasibility checks, using the differential flatness model, and adding the proposed simulator or real-world experiments would take even longer.

In addition, another drawback is the surrogate models' generalizability as mentioned in this paper previously in section IV-A. Since the models are specific to the number of polytopes, waypoints, and environment conditions, if any of those change, a new dataset has to be generated for all vehicles and surrogate models have to be retrained. However, if a new drone is added (while the environment, waypoints, and polytopes stay the same), a new dataset can be generated with subsequent pairwise datasets and surrogate models.

While the paper presents an interesting and mathematically sound framework, we believe there are still significant drawbacks for real-world implementations, and limited applicability for this method.

References

[1] G. Ryou, E. Tal, and S. Karaman, "Cooperative multi-agent trajectory generation with modular bayesian optimization," 2022. [Online]. Available: https://arxiv.org/abs/2206.00726

[2] L. C. Batista da Silva, R. M. Bernardo, H. A. de Oliveira, and P. F. F. Rosa, "Multi-uav agent-based coordination for persistent surveillance with dynamic priorities," in *2017 International Conference on Military Technologies (ICMT)*, 2017, pp. 765–771.

[3] D. Nar and R. Kotecha, "Optimal waypoint assignment for designing drone light show formations," *Results in Control and Optimization*, vol. 9, p. 100174, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666720722000467

[4] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uavs," 2020. [Online]. Available: https://arxiv.org/abs/1908.10541

[5] D. Shishika, J. Paulos, and V. Kumar, "Cooperative team strategies for multi-player perimeter-defense games," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2738–2745, 2020.

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[7] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," 2019.

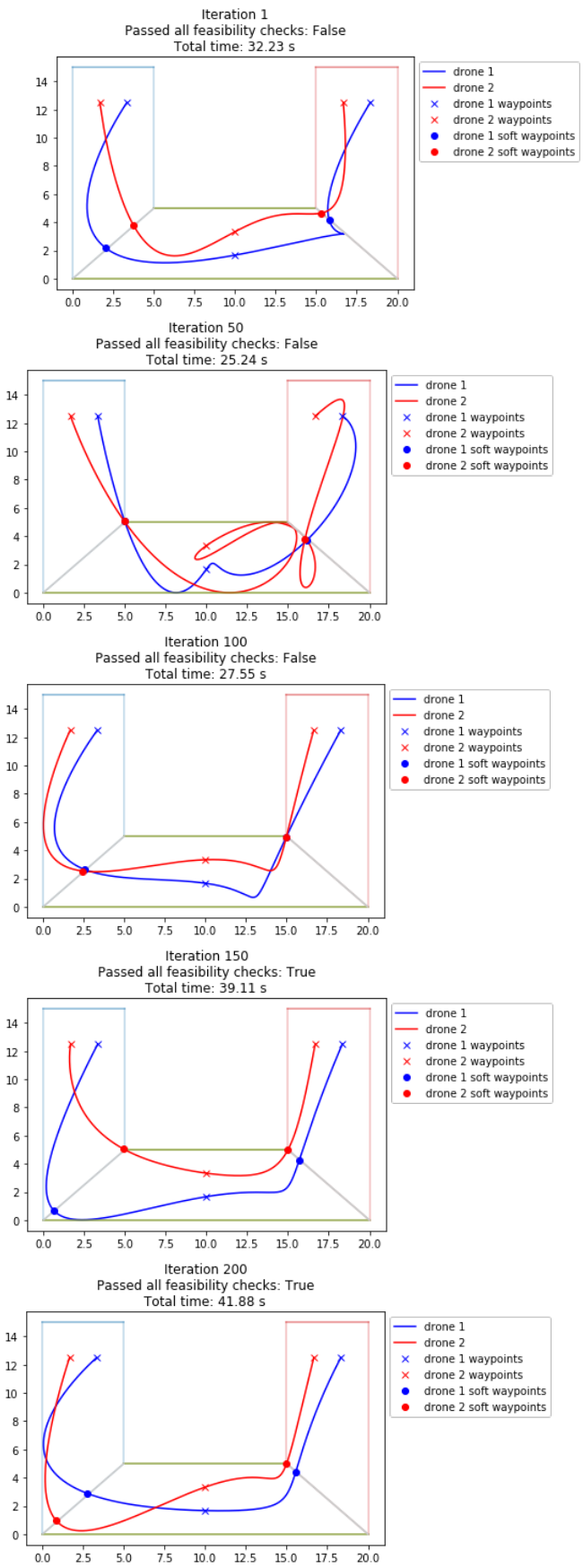

Fig. 5. Trajectories selected by the acquisition function for evaluation during the Bayesian Optimization process.

[8] G. Ryou, E. Tal, and S. Karaman, "Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1352–1369, 2021. [Online]. Available: https://doi.org/10.1177/02783649211033317

[9] J. M. KEIL and J.-R. SACK, "Minimum decompositions of polygonal objects," in *Computational Geometry*, ser. Machine Intelligence and Pattern Recognition, G. T. TOUSSAINT, Ed. North-Holland, 1985, vol. 2, pp. 197–216. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444878069500128