

Cooperative Multi-Agent Trajectory Generation with Modular Bayesian Optimization

Gilhyun Ryou, Ezra Tal, and Sertac Karaman
 Massachusetts Institute of Technology,
 Cambridge, Massachusetts 02139
 Email: {ghryou, eatal, sertac}@mit.edu

Abstract—We present a modular Bayesian optimization framework that efficiently generates time-optimal trajectories for a cooperative multi-agent system, such as a team of UAVs. Existing methods for multi-agent trajectory generation often rely on overly conservative constraints to reduce the complexity of this high-dimensional planning problem, leading to suboptimal solutions. We propose a novel modular structure for the Bayesian optimization model that consists of multiple Gaussian process surrogate models that represent the dynamic feasibility and collision avoidance constraints. This modular structure alleviates the stark increase in computational cost with problem dimensionality and enables the use of minimal constraints in the joint optimization of the multi-agent trajectories. The efficiency of the algorithm is further improved by introducing a scheme for simultaneous evaluation of the Bayesian optimization acquisition function and random sampling. The modular BayesOpt algorithm was applied to optimize multi-agent trajectories through six unique environments using multi-fidelity evaluations from various data sources. It was found that the resulting trajectories are faster than those obtained from two baseline methods. The optimized trajectories were validated in real-world experiments using four quadcopters that fly within centimeters of each other at speeds up to 7.4 m/s.

SUPPLEMENTARY MATERIAL

A video of the experiments is available at <https://youtu.be/rxQiNeXvLTc>. Results and demonstrations also can be found on our project website: <https://aera.mit.edu/projects/MultiDroneModularBayesOpt>.

I. INTRODUCTION

In this paper, we study a multi-agent trajectory optimization problem in which quadcopter vehicles are tasked with traversing a complex environment as fast as possible while avoiding collisions with obstacles and with each other. We consider the problem of designing trajectories for all vehicles involved, minimizing the total time of the mission. Specifically, we are motivated by robotics applications in which multiple vehicles must simultaneously visit certain locations, e.g., to collect synchronized sensor measurements from different viewpoints or to complete a coordinated task or motion in a cooperative manner. These applications add the requirement for agents to pass through multi-agent formation waypoints synchronously.

The problem is an instance of *cooperative* multi-agent planning, in contrast to *non-cooperative* scenarios in which agents have opposing objectives, such as in multi-robot racing [32].

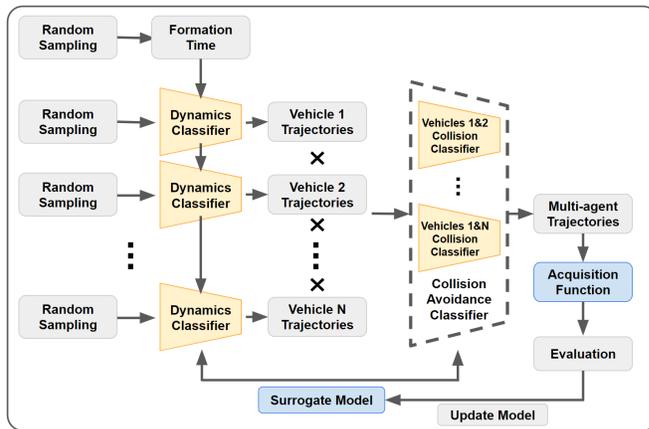


Fig. 1: Overview of our proposed algorithm. The BayesOpt model is composed of multiple Gaussian process models, which represent dynamic feasibility and collision avoidance constraints.

Existing literature has considered cooperative multi-agent motion planning in various contexts, including for unmanned aerial vehicle (UAV) systems. In multi-agent sensor systems, motion planning can be applied to cooperatively gather and share data [28, 31]. Multi-agent systems may also cooperate against a shared adversary, e.g., in target or perimeter defense games where a team of UAVs aims to stop intruders [18, 29].

These multi-agent planning problems have two properties in common that are particularly relevant when trajectories must be as fast as possible. First, collision avoidance between agents should be considered in a spatio-temporal manner, which means that trajectories may intersect as long as vehicles pass through the intersection at different times. Second, vehicles are only required to attain their position within the multi-agent formation at specific points in the trajectory. This implies that—when traveling between these specific waypoints—agents may deviate from the formation in order to achieve more efficient, i.e., faster, trajectories.

Our aim is to generate time-optimal multi-agent trajectories that connect specified start and end points and synchronously attain a sequence of formation waypoints along the way. In order to achieve time optimality, we must explicitly lever-

age the two properties described above. This is challenging because spatio-temporal collision avoidance and formation synchronization require joint and holistic consideration of the agent trajectories. Consequently, the dimension of the input space rapidly increases with the number of agents, leading to prohibitive computational cost. The problem is further complicated by the fact that fast and agile flight in tight formations is intrinsically complex. Unpredictable flight dynamics and subsystem limitations (e.g., in control and actuation) necessitate more sophisticated and computationally costly methods to predict dynamic feasibility of candidate trajectories [27]. These methods are especially needed in time-optimal multi-agent trajectories where vehicles may maneuver in close proximity of each other so that even a small deviation from the anticipated trajectory can result in a collision.

In this paper, we propose an algorithm that enables efficient optimization of multi-vehicle trajectories by alleviating the explosion of computational cost commonly associated with high dimensionality. The algorithm leverages Bayesian optimization (BayesOpt) to efficiently sample the search space and build surrogate models that represent the dynamic feasibility and collision avoidance constraints. While BayesOpt has been applied towards trajectory optimization [27], we present significant innovations that enable application to the high-dimensional cooperative multi-agent planning problem. In addition to the resulting trajectory optimization algorithm, we present several other contributions. First, we present a modular Bayesian optimization architecture that reduces the computational burden of multi-agent optimization. Second, we combine the BayesOpt acquisition function evaluation and random sampling steps to further improve the overall efficiency of Bayesian optimization. Third, we demonstrate that our novel BayesOpt architecture can be applied in a multi-fidelity framework with objective and constraint evaluations from various data sources. Fourth, we present extensive experimental results from the proposed algorithm in six unique environments with obstacles and we evaluate the resulting trajectories in both simulation and real-world flight experiments.

II. PRELIMINARIES

A. Problem Definition

Our goal is to generate time-optimal multi-agent quadrotor trajectories that (i) connect start and end points, (ii) attain specified formation waypoints, and (iii) avoid any collision with obstacles or between vehicles. For each vehicle, a trajectory is represented by a continuous function $p : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^3 \times \mathbb{T}$ —with \mathbb{T} the circle group—that maps time to position and yaw, i.e., $p(t) = [p_r(t)^T \ p_\psi(t)]^T$. Along these trajectories, there are N_f waypoints at which the N_v vehicles must synchronously attain a specified formation. We denote the times at which these formations are attained by $\mathbf{t} = [t_1^{\text{form}} \ \cdots \ t_{N_f}^{\text{form}}]$ and the corresponding position and yaw waypoints for vehicle i by $\tilde{\mathbf{p}}_i^{\text{form}} = [\tilde{p}_{i,1}^{\text{form}} \ \cdots \ \tilde{p}_{i,N_f}^{\text{form}}]$. The total trajectory time for vehicle i is indicated by T_i , so that minimum-time multi-agent planning problem is defined as:

$$\begin{aligned} & \underset{\mathbf{p}, \mathbf{t}, \mathbf{T}}{\text{minimize}} \quad \max_{i=1, \dots, N_v} T_i & (1) \\ & \text{subject to} \\ & p_i(0) = \tilde{p}_i^{\text{start}}, \quad p_i(T_i) = \tilde{p}_i^{\text{end}}, & i = 1, \dots, N_v, \\ & p_i(t_k^{\text{form}}) = \tilde{p}_{i,k}^{\text{form}}, & i = 1, \dots, N_v, \quad k = 1, \dots, N_f, \\ & t_k^{\text{form}} \leq t_{k+1}^{\text{form}}, & k = 1, \dots, N_f - 1, \\ & t_{N_f}^{\text{form}} \leq T_i, \quad p_i \in \mathcal{P}_{T_i}, \quad p_i \in \mathcal{F}_{T_i}, & i = 1, \dots, N_v, \\ & (p_i, p_j) \in \mathcal{F}_{T_i, T_j}, & i, j = 1, \dots, N_v, \quad j > i, \end{aligned}$$

where $\tilde{p}_i^{\text{start}}$ and \tilde{p}_i^{end} are respectively the start and end position and yaw of vehicle i , and $\mathbf{p} = \{p_i\}_{i=1, \dots, N_v}$ and $\mathbf{T} = [T_1 \ \cdots \ T_{N_v}]$. The function space \mathcal{P}_{T_i} is the set of trajectories that satisfy the feasibility constraints over the time interval $[0, T_i]$, i.e., it contains all trajectory functions that the quadrotor can successfully track. Similarly, \mathcal{F}_{T_i} denotes trajectories that satisfy the obstacle avoidance constraints, and \mathcal{F}_{T_i, T_j} denotes trajectory pairs that do not collide.

B. Quadrotor Trajectory Generation

The quadrotor is a highly-agile platform that is capable of tracking challenging maneuvers at high speeds. During such maneuvers, it may exhibit highly nonlinear dynamics, complicating the integration of the feasibility constraints that define \mathcal{P}_{T_i} in trajectory optimization. Popular methods for trajectory planning avoid the complex dynamics constraints by reformulating the optimization problem such that dynamic feasibility is the objective, instead of a constraint [20, 25]. In practice, this is achieved by minimizing high-order derivatives of the trajectory function, particularly, by minimizing the fourth-order derivative of position (i.e., snap) and the second-order derivative of yaw. As these derivatives are related to the angular acceleration of the vehicle, their minimization reduces the required control moment and thereby increases the likelihood that the required motor speeds remain feasible. This approach leads to the following objective function

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

where μ_r and μ_ψ are weighing parameters.

Obstacle avoidance constraints can be incorporated in the trajectory optimization by using polytope constraints to describe their boundaries [3, 5, 27]. Since polytope constraints can be described by linear inequalities, their integration into the optimization formulation does not increase its complexity. In this paper, we utilize the obstacle constraint formulation from [27], which decomposes the obstacle-free space into convex polytopes, resulting in the following minimum-snap optimization:

$$\begin{aligned} & \underset{p}{\text{minimize}} \quad \sigma(p, \sum_{i=1}^m x_i) \\ & \text{subject to} \quad p(0) = \tilde{p}^{\text{start}}, \quad p\left(\sum_{i=1}^m x_i\right) = \tilde{p}^{\text{end}}, \\ & A_i p(t) \leq b_i, \quad \forall t \in \left[\sum_{j=1}^{i-1} x_j, \sum_{j=1}^i x_j \right], & (3) \\ & i = 1, \dots, m, \end{aligned}$$

where the matrix $A_i \in \mathbb{R}^{d_i \times 3}$ and the vector $b_i \in \mathbb{R}^{d_i}$ constrain the i -th trajectory segment to be within a polytope of d_i faces. The vector $\mathbf{x} = [x_1 \ \cdots \ x_m]$ contains the time allocation over the trajectory segments corresponding to these polytopes. By using a piecewise polynomial representation of p , we can effectively represent the set of trajectories that attains these geometric constraints.

By combining the polytope and formation waypoint constraints, we can describe a multi-agent trajectory through N_p polytopes with N_f formation waypoints using $m = N_p + N_f$ trajectory segments per vehicle. The time allocation for a single vehicle can be written as $\mathbf{x} = [x_{e_0} \ \cdots \ x_{e_k} \ \cdots \ x_{e_{N_f+1}}]$, where $e_0 = 1$, $e_{N_f+1} = m$, and e_k is the index of the trajectory segment that ends at the k -th formation waypoint, i.e.,

$$p\left(\sum_{i=1}^{e_k} x_i\right) = \tilde{p}_k^{\text{form}}. \quad (4)$$

For convenience, we denote the function that gives the minimizer trajectory of (3) with (4) for a given time allocation \mathbf{x} as follows:

$$p = \chi(\mathbf{x}, \tilde{\mathcal{F}}), \quad (5)$$

where $\tilde{\mathcal{F}}$ represents $(\tilde{p}^{\text{start}}, \tilde{p}^{\text{end}}, \mathbf{A}, \mathbf{b}, \tilde{\mathbf{p}}^{\text{form}})$ with \mathbf{A} and \mathbf{b} containing respectively all A_i and b_i .

Minimum-snap trajectory generation algorithms commonly employ a two-step process based on (5). First, the minimum-snap trajectory for a (very large) initial guess of the total trajectory time is found, as follows:

$$\begin{aligned} & \underset{\mathbf{x} \in \mathbb{R}_{\geq 0}^m}{\text{minimize}} && \sigma\left(\chi(\mathbf{x}, \tilde{\mathcal{F}}), T\right) \\ & \text{subject to} && T = \sum_{i=1}^m x_i. \end{aligned} \quad (6)$$

Next, the obtained time allocation is scaled down to obtain the minimum-time trajectory, i.e.,

$$\begin{aligned} & \underset{\eta \in \mathbb{R}_{> 0}}{\text{minimize}} && T \\ & \text{subject to} && T = \sum_{i=1}^m \eta x_i, \\ & && \chi(\eta \mathbf{x}, \tilde{\mathcal{F}}) \in \mathcal{P}_T. \end{aligned} \quad (7)$$

The feasibility constraint is typically evaluated using differential flatness of the idealized quadrotor dynamics [20]. Specifically, the flatness transform provides a straightforward manner to obtain the control inputs, i.e., the motor speeds, that correspond to a quadcopter trajectory. The feasibility of the trajectory can then be determined based on the admissibility of its required control inputs.

C. Multi-Agent Trajectories

In addition to the individual dynamics and obstacle avoidance constraints, multi-agent trajectory planning requires collision avoidance constraints for each pair of agents. The problem is challenging because these constraints change as the trajectories of other vehicles are updated.

Trajectory discretization is a popular approach for formulating collision avoidance constraints. By adding minimum-distance constraints between discrete points on each trajectory, the separation between the vehicles can be guaranteed. Since the constraints are in quadratic form, nonlinear optimization is required to solve the problem, e.g., by using a potential field method and iterative updates [15], by utilizing sensitivity-based relaxation [32], or by formulating the mixed-integer linear programming (MILP) that approximates the quadratic constraints with multiple linear constraints and integer variables [21]. Alternatively, collisions can be avoided by utilizing a leader-follower formation [24]. In this approach, a leader trajectory is first formulated, after which the remaining agents are set to follow the leader agent while keeping some formation. Since only a single trajectory optimization is required, this method can relieve the stark increase in complexity, known as the *curse of dimensionality*, caused by the joint optimization of multiple trajectories. The leader-follower approach has been applied to various mobile robotics systems, such as 2D wheeled-robots [22], helicopters [33], and quadrotors [8]. Recently, it was combined with deep reinforcement learning to generate a deep-neural-network-based leader-follower navigation system [4].

In this paper, we compare our trajectory optimization results to two baseline methods: (i) a heuristic formation control scheme, and (ii) the MILP formulation from [21]. In the formation control scheme, vehicle collisions are avoided by enforcing that the vehicles keep a specified formation at all times. We generate the trajectory of the formation center as well as trajectories for two parameters, corresponding to the scale and the rotation of the formation. We specify a formation waypoint k in terms of its center location, the maximum distance bound $\tilde{b}_k^{\text{form}}$ from this center to each of the vehicles, and the formation rotation, i.e., its yaw angle, $\tilde{\psi}_k^{\text{form}}$. The piecewise polynomial trajectory of the formation scale parameter $b^{\text{form}}(t)$ is then generated by solving the following optimization:

$$\underset{b^{\text{form}}}{\text{minimize}} \int_0^{\sum_{i=1}^m x_i} \left\| \frac{d^4 b^{\text{form}}}{dt^4} \right\|^2 \quad (8)$$

$$\begin{aligned} & \text{subject to} && b^{\text{form}}\left(\sum_{i=1}^{e_k} x_i\right) = \tilde{b}_k^{\text{form}}, \quad k = 1, \dots, N_f + 1, \\ & && b^{\text{form}}(t) \leq \max(\tilde{b}_k^{\text{form}}, \tilde{b}_{k+1}^{\text{form}}), \\ & && \forall t \in \left[\sum_{j=1}^{e_k} x_j, \sum_{j=1}^{e_{k+1}} x_j \right], \quad k = 1, \dots, N_f, \\ & && b^{\text{form}}(0) = \tilde{b}_0^{\text{form}}, \\ & && b^{\text{form}}(t) \leq \max(\tilde{b}_0^{\text{form}}, \tilde{b}_1^{\text{form}}), \forall t \in \left[0, \sum_{j=1}^{e_1} x_j \right], \end{aligned}$$

where e_k is the index of the k -th formation waypoint as in (4), and $\tilde{b}_0^{\text{form}}$ and $\tilde{b}_{N_f+1}^{\text{form}}$ refer to the maximum distance bounds at the start and end points. Using the formation scale profile, we generate the trajectory $p^{\text{form}}(t)$ for the formation center using (5) with added separation on the polytope collision avoidance

constraints, as follows:

$$A_j p(t) \leq b_j + b^{\text{form}}(t) E_j, \quad \forall t \in \left[\sum_{i=1}^{j-1} x_i, \sum_{i=1}^j x_i \right], j = 1, \dots, m, \quad (9)$$

where $E_j \in \{0, 1\}^{d_j}$ with its i -th element set to zero if the trajectory passes through the i -th face of polytope j and its remaining elements set to unity. Similar to (8), we connect the formation yaw waypoints ψ_k^{form} with a smooth polynomial $\psi^{\text{form}}(t)$ obtained by minimizing its second-order derivatives. The time allocation is generated using (6) with a modified objective function that includes the constraints from (9). Based on $b^{\text{form}}(t)$, $\psi^{\text{form}}(t)$, and $p^{\text{form}}(t)$, we generate the trajectory for each of the vehicles, and collectively slow down all trajectories until the corresponding required motor speeds remain within the feasible set, similar to (7).

In the MILP-based method, the collision constraints are included in the minimum-snap optimization as mixed-integer constraints. The approach from [21] formulates a MILP with the following vehicle collision avoidance constraints

$$\begin{aligned} |p_{i,k}(t) - p_{j,k}(t)| &\geq d_{\min} - M y_{i,j,k}, \quad k \in \{x, y, z\}, \\ y_{i,j,k} &\in \{0, 1\}, \quad i, j = 1, \dots, N_v, \quad j > i, \\ \sum_{i,j,k} y_{i,j,k} &\leq 3N_v(N_v - 1)/2 - 1, \end{aligned} \quad (10)$$

where M is a large number and d_{\min} is the minimum distance between vehicles. In order to formulate the optimization problem as a linear program, d_{\min} is included as a component-wise constraint. If two vehicles are separated diagonally, collision avoidance can be achieved with a component-wise smaller distance than d_{\min} . Therefore, in our implementation, we run a grid search on d_{\min} and adjust it separately for different obstacle configurations.

D. Bayesian Optimization

Bayesian optimization (BayesOpt) is a class of algorithms that can be used to solve optimization problems with unknown objective or constraint functions that are expensive to evaluate. Evaluation points are selected to model the unknown functions and approach the optimum with maximum efficiency, so that the total number of evaluations is kept to a minimum.

Within the BayesOpt framework, Gaussian process classification (GPC) modeling is widely used to build a *surrogate model* that approximates the unknown constraint functions. Given data points $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with corresponding evaluations $\mathbf{y} = \{y_1, \dots, y_N\}$, GPC assumes a joint Gaussian distribution of the evaluations and the latent variables $\mathbf{f} = [f_1, \dots, f_N]$, and predicts the probability $P(y_* | \mathbf{y}, \mathbf{x}_*, \mathbf{X})$ for a test point \mathbf{x}_* based on the latent variables. These latent variables encode label probabilities for the evaluations, which can be obtained through a mapping onto the probability domain $[0, 1]$, e.g.,

$$\Phi(x) = \int_{-\infty}^x \mathcal{N}(s|0, 1) ds. \quad (11)$$

The latent variables and the hyperparameters of the kernel function are trained by maximizing the marginal likelihood function

$$\begin{aligned} P(\mathbf{y}, \mathbf{f} | \mathbf{X}) &= \prod_{i=1}^N P(y_i | f_i) P(\mathbf{f} | \mathbf{X}) \\ &= \prod_{n=1}^N \mathcal{B}(y_n | \Phi(f_n)) \mathcal{N}(\mathbf{f} | 0, K(\mathbf{X}, \mathbf{X})), \end{aligned} \quad (12)$$

where $\mathcal{B}(x)$ is the Bernoulli likelihood used to formulate $\Phi(f_n)$ as a probability distribution. The covariance between \mathbf{X} and a test point \mathbf{x}_* is modeled with the same covariance kernel, as follows

$$P(\mathbf{f}, f_* | \mathbf{x}_*, \mathbf{X}) = \mathcal{N} \left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \middle| 0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{x}_*) \\ K(\mathbf{x}_*, \mathbf{X}) & K(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right), \quad (13)$$

so that the distribution of the latent variable f_* can be estimated as

$$P(f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int P(f_* | \mathbf{f}, \mathbf{x}_*, \mathbf{X}) P(\mathbf{f} | \mathbf{X}, \mathbf{y}) d\mathbf{f}. \quad (14)$$

The resulting class probability is obtained by

$$P(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int P(y_* | f_*) P(f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}) d\mathbf{f}_*. \quad (15)$$

For more details on GPC and its implementation, the reader is referred to [23].

The BayesOpt *acquisition function* is designed to select the next evaluation point by considering both reducing the uncertainty of the surrogate model and finding the precise optimum of the objective function. Based on the data \mathcal{D} obtained in previous evaluations, the surrogate model can be trained to approximate the unknown function. Next, the approximate optimal solution can be estimated using the trained surrogate model. Each next evaluation point is obtained by solving the following optimization problem:

$$\mathbf{x}_{\text{next}} = \arg \max_{\mathbf{x}} \alpha(\mathbf{x} | \mathcal{D}) \quad (16)$$

where $\alpha(\mathbf{x} | \mathcal{D})$ is the acquisition function that represents the value of an evaluation point \mathbf{x} given the data \mathcal{D} . If the surrogate model is used to approximate an unknown constraint function, the acquisition function can be based on, e.g., the product of the expected objective improvement and the probability of constraint satisfaction [6, 7], or the expected entropy reduction of the distribution over the solution domain that satisfies the constraints [11, 19]. In our proposed algorithm, Bayesian optimization is applied to model the dynamic feasibility and collision avoidance constraints of the multi-agent planning problem.

Multi-fidelity Bayesian optimization combines function evaluations of different fidelity levels. The key idea is that combining cheap low-fidelity evaluations with expensive high-fidelity measurements improves overall efficiency. To incorporate information from multiple sources, the surrogate model must be modified to combine multi-fidelity evaluations, e.g., using a linear transformation to describe the relationship between different fidelity levels [13, 16]. Suppose that we have L fidelity levels, and each level is denoted by $l \in \{l^1, l^2, \dots, l^L\}$,

where l^1 is the level of the lowest-fidelity experiment and l^L is the level of the highest-fidelity experiment, then the relationship between adjacent fidelity levels l^j and l^{j-1} can be modeled as

$$f_{l^j}(\mathbf{x}) = \rho_{l^{j-1}} f_{l^{j-1}}(\mathbf{x}) + \delta_{l^j}(\mathbf{x}), \quad (17)$$

where $f_{l^j}(\mathbf{x})$ and $f_{l^{j-1}}(\mathbf{x})$ are the output distributions of \mathbf{x} for l^j and for l^{j-1} , respectively. The bias distribution δ_{l^j} is independent of $f_{l^{j-1}}, \dots, f_{l^1}$, and the constant $\rho_{l^{j-1}}$ represents the correlation between the output distributions for the two adjacent fidelity levels. Similar to the surrogate model, the acquisition function has to be modified to incorporate multi-fidelity evaluations. In the multi-fidelity Bayesian optimization framework, the acquisition function is used not only to select the next evaluation point, but also its fidelity level, as follows:

$$\mathbf{x}_{\text{next}}, l_{\text{next}} = \arg \max_{\mathbf{x}, l \in \{l^1, \dots, l^L\}} \alpha(\mathbf{x}, l | \mathcal{D}). \quad (18)$$

The acquisition function itself is modified by introducing weights based on the evaluation cost at the different fidelity levels. In practice, high-fidelity evaluations will have smaller weights compared to low-fidelity evaluations. This makes the algorithm less likely to select high-fidelity evaluations, so that the overall cost of the experiments is minimized.

In practice, the acquisition function is often discontinuous or nonlinear and cannot be solved analytically. BayesOpt generates and evaluates (random) candidate solutions to select the acquisition function optimum. The required number of candidate solutions increases exponentially with problem dimension. Consequently, BayesOpt for high-dimensional problems is computationally expensive and often infeasible due to the prohibitive cost of solving the acquisition function. In multi-agent planning, the dimension of the solution space increases linearly with the number of vehicles, leading to an exponential increase in candidate solutions. Recent works on high-dimensional BayesOpt focus on decomposing the high-dimensional space and solving the acquisition function on each subspace separately. For example, [12, 26] assume the acquisition function has additive structure so that the state space can be linearly decomposed into low-dimensional subspaces, and [17, 14] iteratively select active subsets of variables and optimize the acquisition function only on the selected low-dimensional subspace. In our proposed algorithm, we employ a similar approximation on the high-dimensional solution space, where the decomposed acquisition functions correspond to the dynamic feasibility constraints of individual trajectories (first-order) and to the pairwise collision avoidance constraints of pairs of trajectories (second-order).

III. ALGORITHM

We propose a BayesOpt framework to efficiently minimize the total trajectory time $\max_{i=1} T_i$ of a cooperative multi-agent trajectory using multi-fidelity evaluations. We leverage (5) to obtain a mapping from the time allocation over segments \mathbf{x}_i to a smooth minimum-snap trajectory $p_i = \chi(\mathbf{x}_i, \tilde{\mathcal{F}}_i)$ that satisfies the obstacle avoidance and formation waypoint

constraints. This mapping enables us to transform the infinite-dimensional trajectory generation problem to the problem of finding the optimal multi-agent time allocation over segments. Based on this approach, we reformulate the general multi-agent minimum-time planning problem (1), as follows:

$$\begin{aligned} & \underset{\mathbf{x} \in \mathbb{R}_{\geq 0}^{m \times N_v}, \mathbf{T}}{\text{minimize}} \quad \max_{i=1, \dots, N_v} T_i & (19) \\ & \text{subject to} \\ & T_i = \sum_{j=1}^m x_{ij}, & i = 1, \dots, N_v, \\ & \sum_{j=1}^{e_k} x_{i,j} = \sum_{j=1}^{e_k} x_{\tilde{i},j}, & i, \tilde{i} = 1, \dots, N_v, k = 1, \dots, N_f \\ & \chi(\mathbf{x}_i, \tilde{\mathcal{F}}_i) \in \mathcal{P}_{T_i}, & i = 1, \dots, N_v, \\ & \left(\chi(\mathbf{x}_i, \tilde{\mathcal{F}}_i), \chi(\mathbf{x}_j, \tilde{\mathcal{F}}_j) \right) \in \mathcal{F}_{T_i, T_j}, & i, j = 1, \dots, N_v, j > i. \end{aligned}$$

The first constraint in (19) is linear and defines the total trajectory time. The second constraint, also linear, enforces that all formation waypoints are attained synchronously. The two remaining constraints enforce dynamic feasibility of each trajectory and collision avoidance between each pair of trajectories, respectively. As described in Section I, it is challenging to verify these two final constraints. Dynamic feasibility requires that a trajectory can be flown on the actual vehicles and depends on the complex dynamics of fast and agile flight in tight formations. Existing work has already demonstrated that Bayesian optimization can be used to efficiently model this dynamic feasibility constraint [27].

Our major contribution pertains to the handling of the collision avoidance constraint. Since this constraint couples the optimization variables from different vehicles, the trajectory optimization should be solved jointly, resulting in a high-dimensional input domain. As the volume of the data space increases exponentially with dimension, the required amount of data also increases rapidly. This curse of dimensionality prevents the application of standard methods for BayesOpt. In this section, we provide an overview of our proposed modular Bayesian optimization algorithm with particular focus on how we address high dimensionality in the definition of the surrogate model and the acquisition function.

A. Modular Surrogate Model

In multi-agent planning, the number of variables increases proportionally to the number of vehicles. Consequently, surrogate model training and evaluation time increase rapidly as more vehicles are considered. Moreover, modeling performance may suffer, even when provisions are made to handle a large amount of data points, such as by using the inducing points technique as in [10]. For example, in experiments, we noticed that a surrogate model may learn only part of the constraints, e.g., it may represent the dynamic feasibility constraints well but neglect the collision avoidance constraints.

In this work, we propose a modular surrogate model, shown in Fig. 1, to address these challenges. Our model is comprised of several GPC dynamic feasibility constraints and

Algorithm 1: Modular Bayesian optimization with simultaneous acquisition function evaluation

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

1 **subfunction** SampleTraj(i, \mathcal{X}_F):
 2 $\mathcal{X} = \emptyset$
 3 **while** $|\mathcal{X}| < N_1$ **do**
 4 $\mathcal{X}_t \leftarrow$ Randomly sampled N_s points
 5 Rescale \mathcal{X}_t with \mathcal{X}_F
 6 Remove $\mathbf{x} \in \mathcal{X}_t$ s.t. $\tilde{P}_i(y_i = 1|\mathbf{x}, \mathcal{M}_i) < C_1$
 7 $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{X}_t$
 8 **return** \mathcal{X}

9 **function** Modular Bayes-Opt():
 10 **repeat**
 11 $\mathcal{X}_F \leftarrow$ Randomly sampled N_s points
 12 $\mathcal{X} = \mathcal{X}_F$
 13 **while** $|\mathcal{X}| < N_2$ **do**
 14 $\mathcal{X}_t \leftarrow$ SampleTraj(1, \mathcal{X}_F)
 15 **for** $i = 2, \dots, N_v$ **do**
 16 $\mathcal{X}_t \leftarrow \mathcal{X}_t \times$ SampleTraj(i, \mathcal{X}_F)
 17 **for** $j = 1, \dots, i-1$ **do**
 18 Remove $\mathbf{x} \in \mathcal{X}_t$ s.t.
 19 $\tilde{P}_{j,i}(y_{j,i} = 1|\mathbf{x}, \mathcal{M}_{j,i}) < C_2$
 20 $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{X}_t$
 21 $\mathbf{x}, l \leftarrow \arg \max_{\mathbf{x} \in \mathcal{X}, l \in \{1, \dots, L\}} \alpha(\mathbf{x}, l|\mathcal{D})$
 22 Evaluate \mathbf{x} in l -th fidelity experiment
 23 Update dataset $\mathcal{D}_{l,i}, \mathcal{D}_{l,i,j}$
 24 Update the surrogate model
 25 $\mathcal{M}_{1,1}, \dots, \mathcal{M}_{1,N_v}, \mathcal{M}_{1,2}, \dots, \mathcal{M}_{1,N_v-1, N_v}$
 26 **until convergence of solution**
 27 $p_i^* \leftarrow \chi(\mathbf{x}_i^*, \tilde{\mathcal{F}}_i) \forall i = 1, \dots, N_v$
 28 $T^* \leftarrow \max_{i=1, \dots, N_v} \sum_{j=1}^{N_v} x_{ij}^*$
 29 **Output:** $T^*, [p_1^*, \dots, p_{N_v}^*]$

collision models. Each surrogate model \mathcal{M} consists of the latent variables $\mathbf{f} = [f_1 \dots, f_N]$ and the hyperparameters θ of the covariance matrix, such that $\mathcal{M} = (\mathbf{f}, \theta)$. We denote the dataset consisting of trajectories for vehicle i by \mathcal{D}_i . The corresponding surrogate model \mathcal{M}_i , trained with \mathcal{D}_i , approximates the constraints that apply solely to vehicle i , i.e., the obstacle avoidance constraint \mathcal{F}_{T_i} and the dynamic feasibility constraint \mathcal{P}_{T_i} in (1). Similarly, the dataset $\mathcal{D}_{i,j}$ contains multi-agent trajectories for vehicles i and j , and the corresponding surrogate model $\mathcal{M}_{i,j}$, trained with $\mathcal{D}_{i,j}$, approximates the collision avoidance constraint for vehicles i and j , i.e., \mathcal{F}_{T_i, T_j} in (1).

In order to leverage evaluations at L fidelity levels, we further expand the definition of the surrogate model by defining

the model at fidelity level l as

$$\mathcal{M}_l = \{\mathcal{M}_{l,1}, \dots, \mathcal{M}_{l,N_v}, \mathcal{M}_{l,1,2}, \dots, \mathcal{M}_{l,N_v-1, N_v}\}. \quad (20)$$

We then use the multi-fidelity deep Gaussian process (MFDGP) from [2] as covariance kernel function to estimate the multi-fidelity GP prior. Uncertainty quantification computations are accelerated using the inducing points method.

B. Acquisition Function

We use the acquisition function from [27] to consider both *exploration* to improve the surrogate model and *exploitation* to find the optimal solution. In exploration, we select the most uncertain sample near the decision boundary of the classifier [1]. Since the latent function mean approaches zero at the decision boundary, this sample is found as the maximizer of

$$\alpha_{\text{explore}}(\mathbf{x}, l) = - \sum_{i=1}^{N_v} \frac{|\mu_l(\mathbf{x}_i)|}{\sigma_l(\mathbf{x}_i)} - \sum_{i=1}^{N_v-1} \sum_{j=i+1}^{N_v} \frac{|\mu_l(\mathbf{x}_i, \mathbf{x}_j)|}{\sigma_l(\mathbf{x}_i, \mathbf{x}_j)}, \quad (21)$$

where $(\mu_l(\mathbf{x}_i), \sigma_l(\mathbf{x}_i))$ and $(\mu_l(\mathbf{x}_i, \mathbf{x}_j), \sigma_l(\mathbf{x}_i, \mathbf{x}_j))$ are the mean and standard deviation of the l -th fidelity models' posterior distributions $P(f|\mathbf{x}_i, \mathcal{D}_{l,i})$ and $P(f|\mathbf{x}_i, \mathbf{x}_j, \mathcal{D}_{l,i,j})$ obtained from (14).

In exploitation, we utilize expected improvement with constraints (EIC) to quantify the expected effectiveness of a candidate data point based on the product of expected objective improvement and the probability of feasibility [6]. In order to discourage overly optimistic evaluations, we modify EIC to not only consider the probability of success, but also the corresponding variance, as follows:

$$\tilde{P}(y_{l,i} = 1|\mathbf{x}_i) = P(\mu_l(\mathbf{x}_i) - \beta_l \sigma_l(\mathbf{x}_i) \geq 0|\mathbf{x}_i), \quad (22)$$

$$\tilde{P}(y_{l,i,j} = 1|\mathbf{x}_i, \mathbf{x}_j) = P(\mu_l(\mathbf{x}_i, \mathbf{x}_j) - \beta_l \sigma_l(\mathbf{x}_i, \mathbf{x}_j) \geq 0|\mathbf{x}_i, \mathbf{x}_j), \quad (23)$$

where β_l is the penalty weight on the variance. The probability that a set \mathbf{x} of time allocations for all vehicles is feasible is computed as

$$\tilde{P}_l(y = 1|\mathbf{x}) = \left(\prod_{i=1}^{N_v} \tilde{P}_{l,i} \right) \left(\prod_{i=1}^{N_v-1} \prod_{j=i+1}^{N_v} \tilde{P}_{l,i,j} \right), \quad (24)$$

where $\tilde{P}_{l,i} = \tilde{P}(y_{l,i} = 1|\mathbf{x}_i)$ and $\tilde{P}_{l,i,j} = \tilde{P}(y_{l,i,j} = 1|\mathbf{x}_i, \mathbf{x}_j)$. The resulting acquisition function is then given by

$$\alpha_{\text{exploit}}(\mathbf{x}, l) = \begin{cases} \alpha_{EI}(\mathbf{x}) \tilde{P}_l(y = 1|\mathbf{x}), & \text{if } \forall i \neq j, \tilde{P}_{l,i} \geq h_l, \\ & \tilde{P}_{l,i,j} \geq h_l \\ 0, & \text{otherwise} \end{cases} \quad (25)$$

where the bound h_l serves to avoid evaluations that are likely infeasible. Since the objective function is deterministic, so is the expected improvement, given by

$$\alpha_{EI}(\mathbf{x}) = \max_{i=1, \dots, N_v} \left(\sum_j \bar{x}_{ij} \right) - \max_{i=1, \dots, N_v} \left(\sum_j x_{ij} \right) \quad (26)$$

where x_{ij} is a j -th element of i -th vehicle's time allocation \mathbf{x}_i , and $\bar{\mathbf{x}}$ is the current best solution.

Finally, we combine (21) and (25) to obtain

$$\alpha(\mathbf{x}, l) = \begin{cases} \alpha_{\text{exploit}}(\mathbf{x}, l), & \text{if } \exists \mathbf{x} \in \mathcal{X} \text{ s.t. } \alpha_{\text{exploit}}(\mathbf{x}, l) > 0 \\ \alpha_{\text{explore}}(\mathbf{x}, l), & \text{otherwise} \end{cases} \quad (27)$$

where \mathcal{X} is the set of sample trajectories generated in Algorithm 1.

C. Simultaneous Acquisition Function Evaluation and Random Sampling

It is often not possible to find the next data point and its fidelity level by explicitly solving (18). Instead, a typical course in BayesOpt is to first sample a set of candidate data points and then evaluate the acquisition function at all of these data points in order to select the next evaluation point. This method works well in many practical scenarios, but may be problematic when optimizing over a high-dimensional space because the amount of candidate data points required to obtain a reliable result increases exponentially with the dimension. Moreover, the computational burden of each evaluation of the acquisition function, which requires evaluation of (15), increases with the size of the surrogate model and may even surpass the cost of actual evaluation of the data point with regard to the objective and constraints of (19).

In order to improve sampling efficiency, we propose to combine the random sampling and acquisition function evaluation steps. This procedure and the resulting BayesOpt framework are detailed in Algorithm 1. First, we generate a set \mathcal{X}_F of random samples for the time allocation between the formation waypoints. Then, we sample the candidate data points for each vehicle sequentially and obtain the posterior distribution $P(y_i = 1 | \mathbf{x}_i, \mathcal{D}_i)$, which is used in the acquisition function evaluation. When sampling the candidate data points, \mathcal{X}_F is used to rescale the time allocations for each vehicle in order to ensure that the formation waypoints are attained synchronously. Next, the data points with low posterior probability, i.e., those that are least likely to satisfy the dynamic feasibility constraint, are removed from the set of candidate points. Using the remaining data points, we generate pairwise combinations and estimate the distribution $P(y_{i,j} = 1 | \mathbf{x}_i, \mathbf{x}_j, \mathcal{D}_{i,j})$ corresponding to the probability that the collision avoidance constraints are satisfied. Again, we remove data points with low posterior probability. The parameters C_1 and C_2 correspond to the cutoff levels of the rejection sampling steps and are adjusted based on the sample acceptance rate of the previous iteration.

Once the set \mathcal{X} of candidate data points is obtained, the next evaluation points are selected based on (27). At each iteration, the number of evaluation points is adjusted based on the computational cost of the evaluations in the previous iteration. The evaluated points are decomposed into and appended to the datasets \mathcal{D}_i and $\mathcal{D}_{i,j}$ that are used to update the surrogate models \mathcal{M}_i , $\mathcal{M}_{i,j}$. We use the minimum-jerk

approach proposed in [27] to generate smooth samples at all random sampling steps.

D. Initialization and Pre-training

An initial best solution is found by solving (6) and (7) for each vehicle individually. Since the resulting solution does not satisfy the constraint that all vehicles attain the formation waypoints at the same time, we adjust the time allocation for segment j along the trajectory for vehicle i as follows:

$$x_{i,j} \leftarrow x_{i,j} \frac{N_v \sum_{\tilde{j}=e_k}^{e_{k+1}} x_{i,\tilde{j}}}{\sum_{i=1}^{N_v} \sum_{\tilde{j}=e_k}^{e_{k+1}} x_{i,\tilde{j}}}, \quad (28)$$

where $j \in [e_k, e_{k+1})$, so that all vehicles simultaneously attain the next formation waypoint. Finally, we uniformly increase the time allocation to slow down the trajectories until all satisfy the dynamic feasibility constraints and obstacle avoidance constraints again. We note that the resulting initial solution is not necessarily feasible, because it may not satisfy the collision avoidance constraints. However, we found that it does represent a good starting data point around which we sample to obtain the initial data set and build the surrogate models.

We also use the initial best solution for the normalization of the data points. By scaling the time allocation for each trajectory segment using the corresponding value in the initial solution, we maintain solution values of a similar scale, which improves the numerical stability of the training process.

IV. EXPERIMENTAL RESULTS

The proposed algorithm is evaluated in various computational and real-world experiments. First, we focus on generating multi-agent trajectories using single-fidelity Bayesian optimization. We use a simple feasibility check based on the flatness transform, as described in Section II-B. The resulting trajectories, which include challenging obstacle configurations, are compared to results from the two baseline methods described in Section II-C.

Next, we extend the experiments to multi-fidelity Bayesian optimization. Evaluations based on differential flatness now serve as a low-fidelity dataset, while a more realistic 6-DOF flight dynamics simulation is used to generate high-fidelity data. The results demonstrate that our algorithm is capable of learning the direction of the anticipated trajectory-tracking error and considers this direction when modeling the collision avoidance constraints.

In both the single-fidelity and the multi-fidelity experiments, we use the six environments with obstacles shown in Fig. 2 and obtain dynamically feasible multi-agent trajectories that synchronously attain the specified formation waypoints and avoid obstacles and collisions. We validated all six trajectories in real-world flight experiments using four quadcopter vehicles in a motion capture space. A video with animations and recordings of the computational and real-world experiments is included as supplementary material.

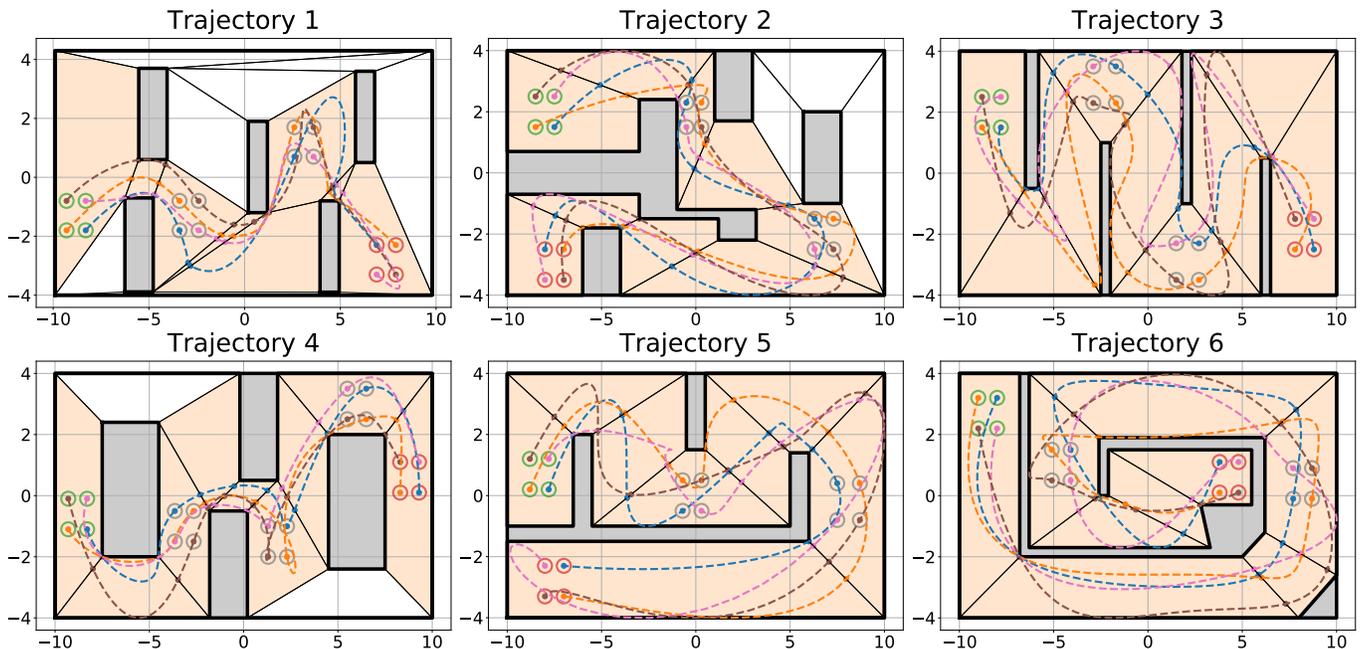


Fig. 2: Single-fidelity trajectory optimization results. Start and end points are indicated by green and red circles, respectively. The synchronized formation waypoints are indicated by gray circles.

A. Single-Fidelity Optimization

In the single-fidelity optimization we verify satisfaction of the collision avoidance constraints based on the planned trajectories and of the dynamic feasibility constraints based on the motor speeds obtained using the differential flatness transform. We set the parameters of the acquisition functions (23) and (25) to $C_{l_1} = 1$, $h_{l_1} = 0.001$ and $\beta_{l_1} = 3.0$ and initialize the adaptive sample rejection thresholds C_1 and C_2 to 0.8. Tuning these parameters mainly affects the computation time and not the actual optimization result, since the parameters mainly affect the efficiency the sampling process. Besides the extreme case where every random sample is rejected, the accepted samples update the surrogate model to arrive at the same optimal solution. At each iteration, these thresholds are adjusted by 1% until an acceptance rate of 0.25 is achieved. The minimum distance between vehicles is set to 40 cm, measured from centroid to centroid. For each of the six environments, we run 200 iterations of Bayesian optimization. At each iteration, the 128 samples with the highest expected improvements based on (27) are selected for evaluation. The relative flight time with regard to the initial best solution (which does not consider collision avoidance) for each iteration is shown in Fig. 3, and the final optimized trajectories are shown in Fig. 2.

The optimized multi-agent trajectory is defined by the time allocation over the trajectory segments for each vehicle. Fig. 2 shows that each trajectory exploits the polytope shapes to avoid collision with other vehicles and obstacles. By increasing the time allocated to a segment, the trajectory can be shifted or a loop can be introduced (e.g., in Trajectory 4) to allow other

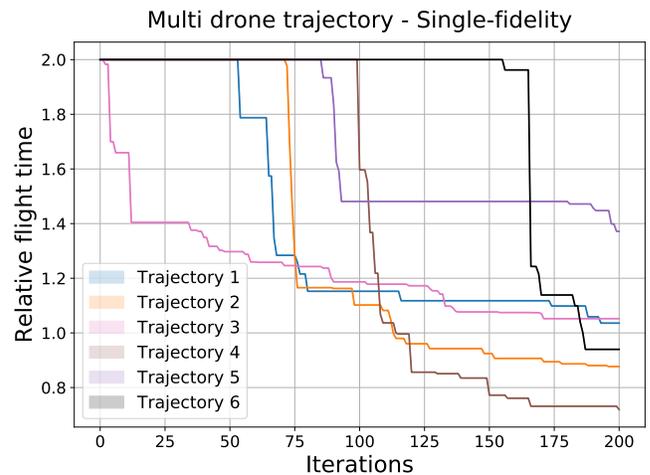


Fig. 3: Relative flight time of single-fidelity optimized trajectories compared to the initial best solution.

vehicles to pass. Similarly, in the later part of Trajectory 3, two of the agents (the brown and pink trajectories) perform large turns, which allows the other two agents to “catch up” so that all agents reach the final formation waypoint at the same time. The wide turns allow the vehicles to “wait” with minimum extra control effort. In addition to the time allocation, we also attempted to optimize the polytope entry and exit points. However, this requires three times more optimization variables and was found to result in inferior results for the same number of iterations.

We compare the optimization results to the two baseline

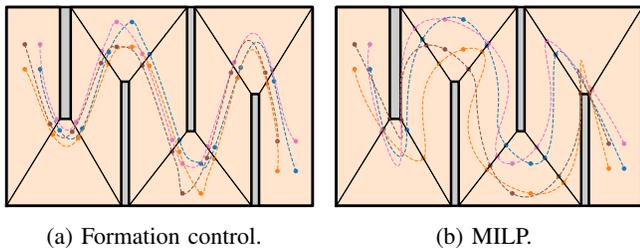


Fig. 4: Baseline results for Trajectory 3.

TABLE I: Trajectory times obtained from formation control, MILP [21], and the proposed modular Bayesian optimization (mBO). Min-snap corresponds to the initial solution obtained from (6) and (7) without considering the collision avoidance constraint.

	Min-snap	Formation	MILP	mBO (Ours)
Traj. 1	5.359 s	14.767 s	Failed	5.548 s
Traj. 2	6.484 s	29.794 s	12.346 s	5.686 s
Traj. 3	7.693 s	14.993 s	9.285 s	8.093 s
Traj. 4	7.256 s	16.375 s	6.951 s	5.221 s
Traj. 5	5.877 s	16.050 s	9.608 s	8.061 s
Traj. 6	8.206 s	34.500 s	11.083 s	7.713 s

methods described in Section II-C: formation control and the MILP-based algorithm from [21]. The formation control method reliably generates feasible trajectories, but the resulting trajectories slow down significantly while the formation is scaling or rotating. In contrast, the MILP-based algorithm generates typically faster trajectories. However, since the MILP requires a large number of optimization variables, the optimization algorithm often fails to find a solution. Table I presents the trajectory times obtained using the baseline methods and the proposed algorithm. Figure 4 clearly shows how the baseline algorithms result in different trajectories. When comparing Fig. 4b to the corresponding trajectory in Fig. 2, we notice that our proposed algorithm is able to generate a faster trajectory because of its capability to incorporate less conservative collision avoidance constraints.

B. Multi-Fidelity Optimization

For the multi-fidelity experiments, the motor speed check based on differential flatness serves as low-fidelity evaluation and a more realistic 6-DOF flight dynamics simulation is used for high-fidelity evaluations. We utilize the open-source multicopter dynamics and inertial measurement simulation by [9] with the trajectory-tracking controller by [30]. For the high-fidelity evaluations, we set the maximum trajectory tracking error to 5 cm and the minimum distance between vehicles to 40 cm. The initial best solution is obtained by slowing down the resulting trajectory from the single-fidelity optimization until the trajectory-tracking error bound is satisfied. The low-fidelity dataset and the surrogate model are initialized with the dataset generated from the single-fidelity optimization. We again use $h_{l_2} = 0.001$ and $\beta_{l_2} = 3.0$ for the acquisition functions (23) and (25), but now set $C_{l_2} = 10$ to consider the increased

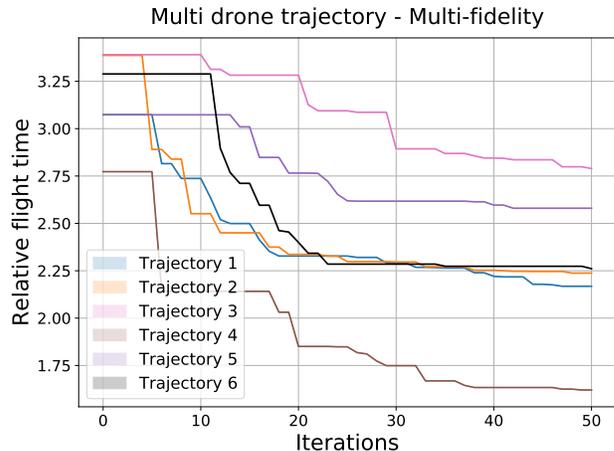


Fig. 5: Relative flight time of multi-fidelity optimized trajectories compared to the initial best solution.

evaluation cost. For each trajectory, we run 50 iterations of Bayesian optimization. At high-fidelity iterations at most four samples are evaluated, while low-fidelity iterations consider 64 samples. Figure 6 contains the resulting trajectories along with the trajectories obtained from the single-fidelity optimization. The relative flight time for each iteration is presented in Fig. 5.

When examining Fig. 6, we observe that the multi-fidelity optimized trajectories significantly deviate from the single-fidelity trajectories. Still, both sets of trajectories exhibit similar features. For instance, in both cases, the orange vehicle in Trajectory 4 makes a loop to wait for the other vehicles. Figure 7 provides an additional detailed comparison of part of Trajectory 1. It can be seen that the single-fidelity trajectory of the blue vehicle takes a wide turn towards the top-right corner of the inset in order to avoid the orange vehicle. In contrast, the corresponding multi-fidelity trajectory is much closer to the orange vehicle. The multi-fidelity optimizer exploits the anticipated tracking error, which moves the blue vehicle away from the orange vehicle, to avoid the wide turn in the blue trajectory.

C. Flight Experiments

The trajectories obtained by the multi-fidelity optimization were further tested in real-world flight experiments in the motion capture space shown in Fig. 8. Figure 9 shows two of the four quadcopter vehicles used in the flight experiments, as well as a tape measure that illustrates their close proximity at the minimum distance of 40 cm between centroids. The minimum separation between vehicles is not more than a few centimeters, while the trajectories reach speeds up to 7.4 m/s.

The flight experiments demonstrate that the optimized trajectories can be flown on the real-world quadcopter vehicles. A video of the experiments is included as supplementary material. Due to the very small minimum distance, even a tiny discrepancy between the anticipated and actual trajectory-tracking error can result in a collision. Indeed, we observed

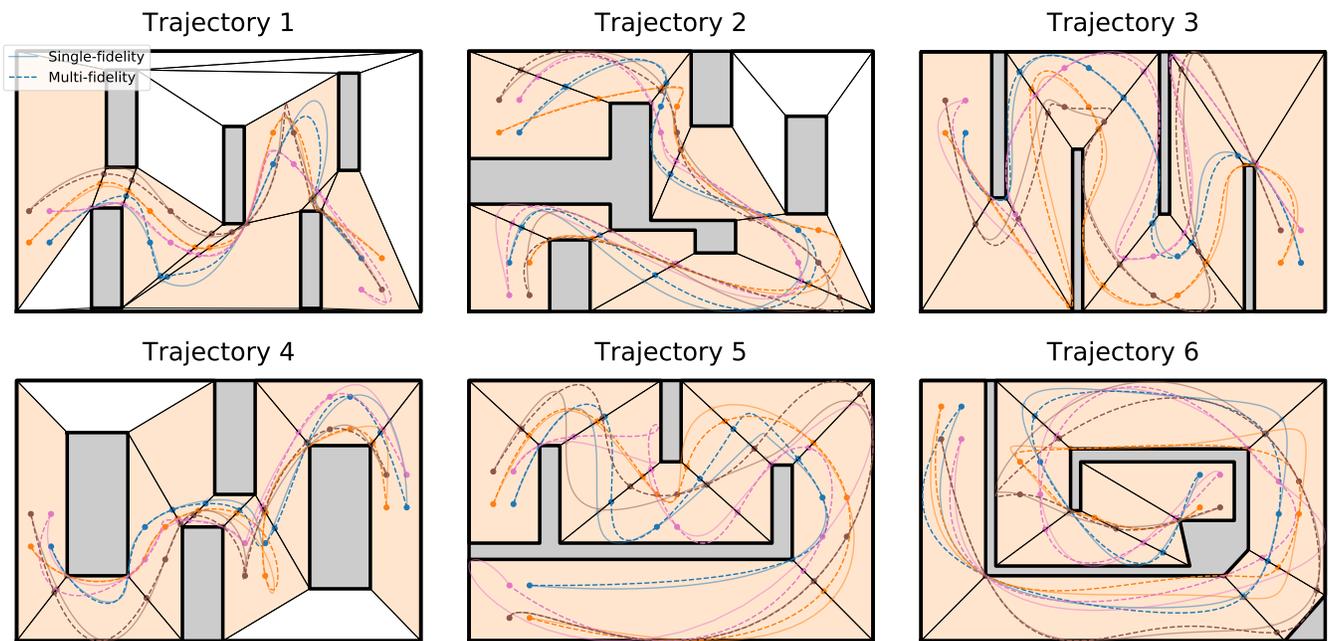


Fig. 6: Single-fidelity (solid curve) and multi-fidelity (dashed curve) trajectory optimization results.

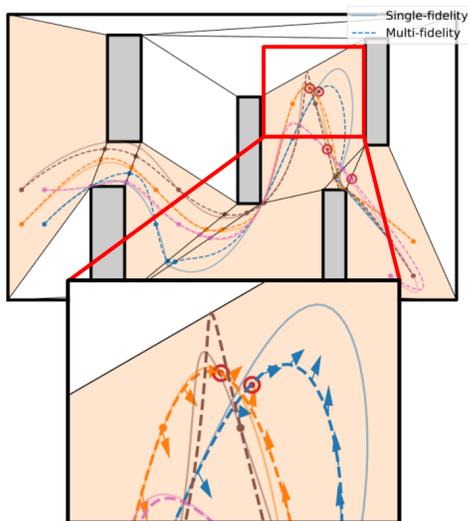


Fig. 7: Single-fidelity (solid curve) and multi-fidelity (dashed curve) results for Trajectory 1. The red circles indicate the positions where the vehicles are closest to each other. The arrows indicate the tracking error, which is scaled 20 times. Since the direction of tracking error moves the blue trajectory away from the orange trajectory, the two trajectories can be placed closer together.

contact between two of the vehicles in Trajectory 1 and in Trajectory 2. Despite the contact, all vehicles were able to continue flying and successfully completed every trajectory. Figure 10 shows the actual and anticipated tracking error around the point of contact in Trajectory 2. It can be seen that the anticipated tracking error of the blue vehicle is in the

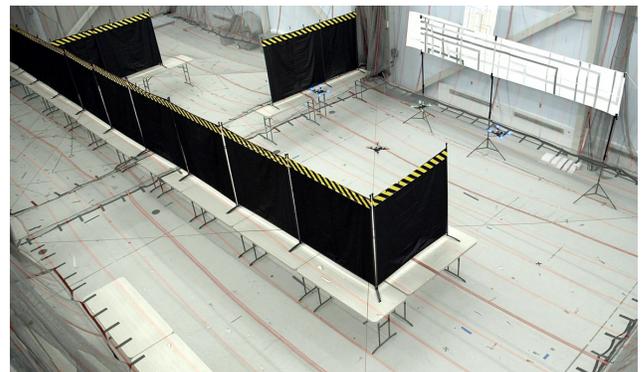


Fig. 8: Environment for multi-agent flight experiments.

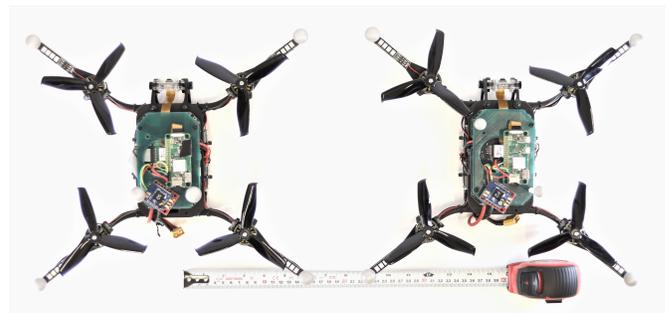


Fig. 9: Quadcopter vehicles used for multi-agent flight experiments. Tape measure indicates 40 cm.

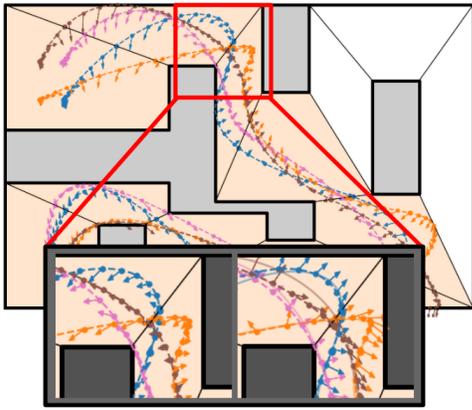


Fig. 10: Tracking error in real-world flight for Trajectory 2. The arrows indicate the tracking error, which is scaled 20 times. The inset shows the actual (left) and anticipated (right) tracking error.

wrong direction, resulting in contact with the brown vehicle. The discrepancies between trajectory tracking in simulation and in real-world flight could potentially be addressed by incorporating actual flight experiments in the multi-fidelity optimization [27].

V. CONCLUSION

We have presented a novel modular Bayesian optimization algorithm to generate cooperative multi-agent trajectories subject to dynamic feasibility, obstacle avoidance, collision avoidance, and formation waypoint constraints. The BayesOpt surrogate model is decomposed into multiple Gaussian process classifiers in order to alleviate the rapid increase in computational cost with increasing dimensionality. Each module of the surrogate model approximates either individual dynamic feasibility constraints or pair-wise collision avoidance constraints. By combining the acquisition function evaluation and the random sampling steps in BayesOpt, we further improve the overall efficiency of the Bayesian optimization. The resulting algorithm is validated through simulation and real-world flight experiments in six unique environments with obstacles.

Future work may focus on resolving several limitations of the current algorithm. Although we reduce the computation time with the modular structure and the efficient sampling method, the current implementation of our algorithm requires between two and three hours to optimize a multi-vehicle trajectory. The main bottleneck is the updating time of the surrogate model, as the algorithm trains each module sequentially due to GPU memory limitations. We expect that parallelization of the training process will reduce the computation time.

Another limitation of the algorithm is that it cannot check whether a feasible solution exists before running the optimization procedure. While the algorithm can find good solutions even when the initial trajectory is infeasible, we have noticed that it often fails to find any feasible solution if there is a narrow corridor or—equivalently—if a large minimum distance between vehicles is required. If a feasible solution exists,

these cases could potentially be addressed by adjusting the training parameters. This motivates us to consider a feasibility check that can establish the likely existence of a solution based on heuristic methods. We conjecture that analyzing how the trajectory generation algorithm utilizes the shape of the polytopes may provide a way to estimate the existence of the solution. Since the algorithm exploits the polytope shapes to shift or slow down the trajectory, the conditions for the optimal polygonal decomposition could inform whether a feasible solution exists.

ACKNOWLEDGMENTS

This work was partly supported by the Army Research Office through grant W911NF1910322.

REFERENCES

- [1] Francisco Sahli Costabal, Paris Perdikaris, Ellen Kuhl, and Daniel E Hurtado. Multi-fidelity classification using Gaussian processes: accelerating the prediction of large-scale computational models. *Computer Methods in Applied Mechanics and Engineering*, 357:112602, 2019.
- [2] Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. Deep Gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320*, 2019.
- [3] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for UAVs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49. IEEE, 2015.
- [4] Ankur Deka, Wenhao Luo, Huao Li, Michael Lewis, and Katia Sycara. Hiding Leader’s Identity in Leader-Follower Navigation through Multi-Agent Reinforcement Learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4769–4776. IEEE, 2021.
- [5] Fei Gao, William Wu, Yi Lin, and Shaojie Shen. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351. IEEE, 2018.
- [6] Jacob R. Gardner, Matt J. Kusner, Zhixiang Eddie Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning (ICML)*, pages 937–945. PMLR, 2014.
- [7] Michael Adam Gelbart. *Constrained Bayesian optimization and applications*. PhD thesis, Harvard University, 2015.
- [8] Khaled A Ghamry and Youmin Zhang. Formation control of multiple quadrotors based on leader-follower method. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1037–1042. IEEE, 2015.
- [9] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. FlightGoggles: photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. In *2019 IEEE/RSJ In-*

- International Conference on Intelligent Robots and Systems (IROS)*, pages 6941–6948. IEEE, 2019.
- [10] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 351–360. PMLR, 2015.
- [11] José Miguel Hernández-Lobato, Michael Gelbart, Matthew Hoffman, Ryan Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. In *International conference on machine learning (ICML)*, pages 1699–1707. PMLR, 2015.
- [12] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning (ICML)*, pages 295–304. PMLR, 2015.
- [13] Marc C. Kennedy and Anthony O’Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.
- [14] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning (ICML)*, pages 3429–3438. PMLR, 2019.
- [15] Paul Ladinig, Bernhard Rinner, and Stephan Weiss. Time and Energy Optimized Trajectory Generation for Multi-Agent Constellation Changes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7400–7406. IEEE, 2021.
- [16] Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5):365–386, 2014.
- [17] Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional Bayesian optimization using dropout. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2096–2102, 2017.
- [18] Li Liang, Fang Deng, Zhihong Peng, Xinxing Li, and Wenzhong Zha. A differential game for cooperative target defense. *Automatica*, 102:58–71, 2019.
- [19] Alonso Marco, Dominik Baumann, Philipp Hennig, and Sebastian Trimpe. Classified Regression for Bayesian Optimization: Robot Learning with Unknown Penalties. *arXiv preprint arXiv:1907.10383*, 2019.
- [20] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *International Conference on Robotics and Automation (ICRA)*, pages 2520–2525. IEEE, 2011.
- [21] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 477–483. IEEE, 2012.
- [22] Zhiqiang Miao, Yun-Hui Liu, Yaonan Wang, Guo Yi, and Rafael Fierro. Distributed estimation and control for leader-following formations of nonholonomic mobile robots. *IEEE Transactions on Automation Science and Engineering*, 15(4):1946–1954, 2018.
- [23] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.
- [24] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.
- [25] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *International Symposium on Robotics Research (ISRR)*, pages 649–666. Springer, 2016.
- [26] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics (AISTATS)*, pages 298–307. PMLR, 2018.
- [27] Gilhyun Ryou, Ezra Tal, and Sertac Karaman. Multifidelity black-box optimization for time-optimal quadrotor maneuvers. *The International Journal of Robotics Research (IJRR)*, 40(12-14):1352–1369, 2021.
- [28] Mac Schwager, Brian J Julian, and Daniela Rus. Optimal coverage for multiple hovering robots with downward facing cameras. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3515–3522. IEEE, 2009.
- [29] Daigo Shishika, James Paulos, and Vijay Kumar. Cooperative team strategies for multi-player perimeter-defense games. *IEEE Robotics and Automation Letters (RAL)*, 5(2):2738–2745, 2020.
- [30] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. *IEEE Transactions on Control Systems Technology*, 29(3):1203–1218, 2021.
- [31] Yulun Tian, Katherine Liu, Kyel Ok, Loc Tran, Danette Allen, Nicholas Roy, and Jonathan P How. Search and rescue under the forest canopy using multiple UAVs. *The International Journal of Robotics Research (IJRR)*, 39(10-11):1201–1221, 2020.
- [32] Zijian Wang, Riccardo Spica, and Mac Schwager. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pages 225–238. Springer, 2019.
- [33] Ben Yun, Ben M Chen, Kai-Yew Lum, and Tong H Lee. A leader-follower formation flight control scheme for UAV helicopters. In *2008 IEEE International Conference on Automation and Logistics*, pages 39–44. IEEE, 2008.