2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Multi-Fidelity Reinforcement Learning for Minimum Energy Trajectory Planning

Luke de Castro[1], Gilhyun Ryou[1], Hyungseuk Ohn[2], and Sertac Karaman[1]

***Abstract*— Modeling the energy consumption of a quadrotor involves complex electrical and physical dynamics, making it difficult to optimize. To address this challenge, this paper presents a multi-fidelity Gaussian process (MFGP) method that efficiently learns an accurate energy prediction model by combining many low-fidelity samples from a simple motor model with a few computationally expensive samples from a numerical battery simulation. We present extensive sample-efficiency experiments, demonstrating that a single-fidelity model often needs 10 times more high-fidelity data to match the accuracy achieved by the MFGP. The energy prediction model is then applied to a reinforcement learning (RL) agent, providing a reward signal to a minimum energy planning policy. The RL policy generates more energy efficient trajectories than those found by the minimum snap baseline method, achieving an average 3.6% energy reduction.**

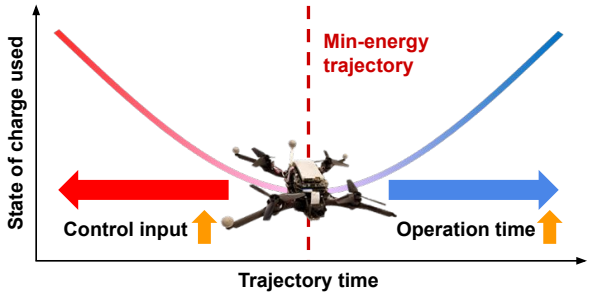


Fig. 1: The main challenge in minimum energy trajectory planning involves balancing minimizing operating time with average power consumption. The proposed method provides an accurate power consumption model to precisely identify the optimal balance between these two objectives.

I. Introduction

Energy-efficient trajectory planning is a critical component in autonomous vehicles and mobile robots because it ensures the most effective use of the limited energy budget. In numerous mobile robotics applications, energy efficiency is closely linked to the speed profile. Energy is defined as the product of power and operation time. As such, one way to reduce energy consumption is to shorten the total trajectory time. However, such trajectories can be risky and demand a higher actuator utilization, leading to increased power use. Conversely, slow trajectories can consume more energy due to the static power required for operating—UAVs need power for hovering, and cars consume energy to keep the engine running. The balance between minimizing average power and total operating time is illustrated in Figure 1. This paper aims to identify the optimal balance between speed extremes, with a specific focus on quadrotor vehicles.

To generate energy-efficient trajectories, several works incorporate an energy model, either aerodynamic or electrical, as an objective function of non-linear trajectory optimization. [1] employs an aerodynamics-based energy model to evaluate the energy of elementary quadrotor maneuvers. [2], [3] model both the aerodynamics and electrical systems to generate a series of trajectory primitives through nonlinear optimization. [4] instead studies different polynomial trajectory classes to find the most energy efficient choice for a set of one dimensional trajectories. It concludes that minimum snap trajectories, which produce a smooth fourth-order polynomial, are the most energy efficient compared to lower order polynomials. [5] conducts more extensive studies over a polynomial trajectory class connecting randomly generated trajectories in 3-D, which shows that the minimum acceleration trajectories are more energy efficient than snap and jerk minimization under a fixed time allocation.

Existing approaches often simplify energy modeling to manage a trade-off between computational efficiency and optimization complexity. Such simplified models cannot fully account for critical aspects of battery usage—like voltage drops, changes in internal resistance, and degradation—all of which are crucial in practical applications. Moreover, these approaches often necessitate predetermining the final trajectory time, focusing solely on optimizing collocation points to reduce the complexity of the optimization process. However, these approaches overlook the fact that operating time significantly influences energy optimization. Furthermore, the non-linear relationship between energy consumption and speed implies that a trajectory considered optimal at a given operating time might not remain optimal if the time varies.

Our main contribution is a multi-fidelity technique to train an energy prediction model efficiently, as well as a reinforcement learning (RL) framework to develop a planning policy based on this prediction model. The key approach is to efficiently leverage complex and realistic simulations to train the energy prediction model, thereby avoiding the need to oversimplify battery models. It's noteworthy that the multi-fidelity method has been applied to achieve time-optimal trajectories in our previous work [6]. Building upon this approach, this paper studies the modeling of quadrotor energy consumption ahead of time, which could provide additional insights for mission planning while also enabling efficient trajectory optimization.

[1]Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 {lukedc, ghryou, sertac}@mit.edu
[2]Hyundai Motor Company hsohn@hyundai.com

- **Energy prediction model**: We train the energy model with data from a realistic battery simulation to accurately capture complex battery performance. The number of training data points is minimized by integrating a simple power simulation (low-fidelity model) using a multi-fidelity Gaussian process (MFGP) model. This approach enables the energy prediction model to achieve a low estimation error with just a few high-fidelity samples, significantly outperforming a single-fidelity model trained without this low-fidelity augmentation.
- **Training planning policy**: The planning policy is trained to maximize the energy savings estimated by the energy prediction model. By employing the energy prediction model as a reward function, we circumvent the need for numerous expensive evaluations for RL training. The trained policy model generates trajectories that consume 3.6 % less energy compared to baseline minimum-snap trajectories, which were previously considered highly energy-efficient. In contrast, when the policy is trained for minimum time, it produces faster trajectories but at the cost of a 60 % increase in battery consumption.

II. Preliminaries

A. Min-snap Trajectory Optimization

Minimum snap trajectory planning is a popular choice for quadrotors [7]. The method produces a fourth-order polynomial in position and a second order polynomial for yaw angle. As the fourth-order derivative of position represents the rate of change of the vehicle's state, minimizing it produces trajectories that require less rapid motor adjustments, often regarded as energy-efficient flight paths for quadrotors [4]. For a sequence of m-waypoints $\tilde{p}^i$, each consisting of a prescribed position $\tilde{p}^i_r \in \mathbb{R}^3$ and yaw angle $\tilde{p}^i_\psi \in \mathbb{S}^1$ ($\tilde{\mathbf{p}} = \begin{bmatrix}\tilde{p}^0 \cdots \tilde{p}^m\end{bmatrix}$), this approach models the trajectory using piecewise continuous polynomials. These polynomials map time to position and yaw, i.e. , $\mathbf{p}(t) = \begin{bmatrix}p_r(t)^\intercal & p_\psi(t)\end{bmatrix}^\intercal$ ($p_r(t) \in \mathcal{C}^4, p_\psi(t) \in \mathcal{C}^2$), and each polynomial segment connects two adjacent waypoints. $\mathbb{S}^1$ denotes the circular angle space, and $\mathcal{C}^n$ is the differentiability class where its n-th order derivatives exist and are continuous.

The coefficients of the polynomial trajectory are determined by solving the following optimization problem:

$$\begin{aligned} &\underset{p=[p_r,p_\psi],\mathbf{x}\in\mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad \sigma\left(\mathbf{x}, p\right) \\ &\text{subject to } \; p(0) = \tilde{p}^0, \;\; p\left(\textstyle\sum_{j=1}^{i} x_j\right) = \tilde{p}^i, \; i = 1, \ldots, m, \\ &\qquad\qquad p \in \mathcal{P} \end{aligned} \tag{1}$$

where

$$\sigma(\mathbf{x}, p) = \int_0^{\sum_{i=1}^m x_i} \left\| \frac{d^4 p_r}{d^4 t} \right\|^2 + \left(\frac{d^2 p_\psi}{d^2 t} \right)^2 dt. \tag{2}$$

$\mathcal{P}$ represents the set of dynamically feasible trajectories, i.e., trajectories that require admissible motor speed commands. x_i, the pivotal optimization variable, denotes the time allocation between two consecutive waypoints $\tilde{p}_{i-1}$ and $\tilde{p}_i$ ($\mathbf{x} = \begin{bmatrix}x_1 \cdots x_m\end{bmatrix}$), and is exclusively determined through non-linear optimization.

This method consists of three components to efficiently solve the optimization problem: 1) Inner-loop spatial optimization that uses quadratic programming to derive polynomial coefficients based on the time allocation between waypoints:

$$\begin{aligned} \chi(\mathbf{x}, \tilde{\mathbf{p}}) = &\underset{p=[p_r,p_\psi]}{\text{argmin}} \quad \sigma\left(\mathbf{x}, p\right) \\ \text{subject to } \; p(0) = \tilde{p}^0, \;\; &p\left(\textstyle\sum_{j=1}^{i} x_j\right) = \tilde{p}^i, \; i = 1, \ldots, m, \end{aligned} \tag{3}$$

2) Outer-loop temporal optimization via non-linear programming that minimizes the output of the inner-loop quadratic programming and determines the time allocation ratio, $\tilde{\mathbf{x}}$:

$$\underset{\tilde{\mathbf{x}}\in\mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad \sigma\left(\chi(\tilde{\mathbf{x}}, \tilde{\mathbf{p}}), \tilde{\mathbf{x}}\right) \;\; \text{subject to } \sum\nolimits_{i=1}^{m} \tilde{x}_i = 1, \tag{4}$$

3) Line search optimizing over a scalar α to generate the feasible trajectory with the lowest energy, given an energy evaluation function f_e:

$$\underset{\alpha\in\mathbb{R}_{>0}}{\text{minimize}} \; f_e(\alpha\tilde{\mathbf{x}}, \tilde{\mathbf{p}}), \;\; \text{subject to } \; \chi(\alpha\tilde{\mathbf{x}}, \tilde{\mathbf{p}}) \in \mathcal{P}. \tag{5}$$

Here, the actual time allocation is defined as the product of α and the time allocation ratio: $\mathbf{x} = \alpha\tilde{\mathbf{x}}$.

When a vehicle starts and ends in a stationary state, with zero velocity and acceleration, uniformly scaling time allocations does not alter the trajectory's shape but shifts control commands away from the default stationary control commands [7]. This method can be computationally expensive, depending on the f_e. This optimization method serves as the baseline for comparing the energy consumption against our resulting trained policy.

B. Multi-fidelity Gaussian Process

In this paper, we employ the Multi-fidelity Gaussian Process (MFGP) to efficiently model the energy function based on sparse multi-fidelity evaluations. Given a collection of data points $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_N\}$ with their associated evaluation outcomes from the l-th fidelity level $\mathbf{y}_l = \{y_{l,1}, \cdots, y_{l,N}\}$, the prior Gaussian distribution is modeled using a multi-fidelity covariance kernel. The key idea of the multi-fidelity covariance kernel $K_l(\mathbf{Z}, \mathbf{Z})$ is utilizing cheap low-fidelity evaluations to efficiently model the high-fidelity measurements. The relationship between the distributions of different fidelity levels is represented by the nonlinear space-dependent transformation proposed in [8]. An element of the l-th covariance kernel $k_l(z, z')$ is estimated as

$$\begin{aligned} k_l(z, z') = k_{l,corr}(z, z')(&\sigma^2_{l,linear}\, f_{l-1}(z)^T f_{l-1}(z') \\ &+ k_{l,prev}(f_{l-1}(z), f_{l-1}(z'))) + k_{l,bias}(z, z'), \end{aligned} \tag{6}$$

where f_{l-1} is the Gaussian process estimation from the preceding fidelity level, $\sigma_{l,linear}$ is a constant scaling the linear covariance kernel, and $k_{l,prev}$, $k_{l,corr}$ and $k_{l,bias}$ represent the covariance with the preceding fidelity, the space-dependent correlation function and the bias function,

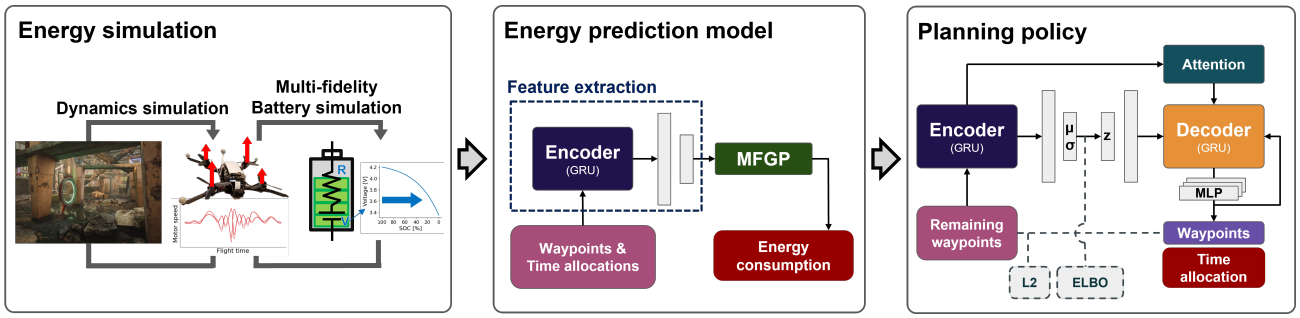


Fig. 2: Overview of the proposed algorithm: The approach begins by modeling quadrotor energy dynamics based on battery characteristics. A multi-fidelity Gaussian process model approximates this battery simulation, improving computational efficiency. This energy model then provides the reward signal for a reinforcement learning policy, generating minimum energy trajectories.

respectively. Based on the covariance matrix with the multi-fidelity kernel, the likelihood is modeled as

$$P(\mathbf{y}_l|\mathbf{Z}) = \mathcal{N}(\mathbf{y}_l|0, K_l(\mathbf{Z},\mathbf{Z})), \tag{7}$$

and the prediction on the l-th fidelity level over the new data points is obtained by marginalizing the training data points as follows:

$$P(y_*|\mathbf{z}_*, \mathbf{Z}_l, \mathbf{y}_l) = \int P(y_*|\mathbf{z}_*, \mathbf{Z}_l, \mathbf{y}_l) P(\mathbf{y}_l|\mathbf{Z}_l) d\mathbf{y}_l. \tag{8}$$

MFGP is further accelerated with the inducing points method [9], [10], [11], which approximates the distribution $P(\mathbf{y}_l|\mathbf{Z})$ by introducing a variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$. The hyperparameters $\mathbf{m}, \mathbf{S}$ represent the mean and covariance of the inducing points $\mathbf{u}$. The method minimizes the following variational lower bound as a loss function to determine the inducing points and maximize the marginal likelihood $P(\mathbf{y}_l|\mathbf{Z}_l)$:

$$\mathcal{L}_{\mathcal{M}}(\mathbf{y}, \mathbf{Z}) = -\mathbb{E}_{q(\mathbf{u})}\left[\log P(\mathbf{y}|\mathbf{u})\right] + D_{\mathrm{KL}}\left[q(\mathbf{u}) \| P(\mathbf{u}|\mathbf{Z})\right]. \tag{9}$$

$\mathcal{M} := (\theta, \mathbf{m}, \mathbf{S})$ is the set of all hyperparameters, and θ is the hyperparameters of the Gaussian process kernel. These hyperparameters of the MFGP are determined by minimizing the sum of the variational lower bounds across all fidelity levels:

$$\mathcal{L}_{\mathrm{MFGP}} = \sum_{l=1}^{L} \mathcal{L}_{\mathcal{M}_l}(\mathbf{y}_l, \mathbf{Z}_l). \tag{10}$$

The MFGP is implemented with GPyTorch [12] based on the work presented in [13].

III. Methods

The objective of the proposed method is to develop planning policy that generates minimum-energy trajectory. For a given waypoint sequence $\tilde{\mathbf{p}}$, we formulate the trajectory optimization problem as follows:

$$\underset{\mathbf{x}}{\text{minimize}}\ f_e(\mathbf{x}, \tilde{\mathbf{p}}), \quad \text{subject to}\ \ \chi(\mathbf{x}, \tilde{\mathbf{p}}) \in \mathcal{P}. \tag{11}$$

Our optimization variables are the time allocations between waypoints, representing the traversal times. A piecewise polynomial trajectory is derived from the time allocation through the quadratic programming of the minimum snap method. $f_e(\mathbf{x}, \tilde{\mathbf{p}})$ in (11) refers to the energy estimate of the polynomial trajectory obtained from a time allocation $\mathbf{x}$ and sequence of waypoints $\tilde{\mathbf{p}}$.

We introduce a multi-fidelity reinforcement learning (MFRL) approach for training a planning policy that determines energy optimal time allocations for a given sequence of waypoints. Accurate energy consumption estimation necessitates complex simulations, leading to extended evaluation times. To reduce the number of necessary energy consumption estimations for training, we create a proxy model for energy estimation, denoted $\hat{f}_e$, using a Gaussian process. We employ a multi-fidelity kernel within the Gaussian process, complemented by a hierarchical evaluation method, to enhance the efficiency of model construction. Subsequently, the planning policy is trained with RL, aiming to minimize energy consumption estimates obtained from the Gaussian process model.

A. Estimation of Energy Consumption

We use two methods, differing in accuracy and computational demand, to estimate energy consumption for planned trajectories. For given sets of time allocations and waypoint sequences, both models calculate the energy consumption of the piecewise polynomial trajectory derived from (3). Due to differences in fidelity, the optimal trajectories identified by each model may vary. By leveraging both models, our objective is to efficiently determine optimal minimum-energy trajectories for high-fidelity evaluation, balancing accuracy with computational cost.

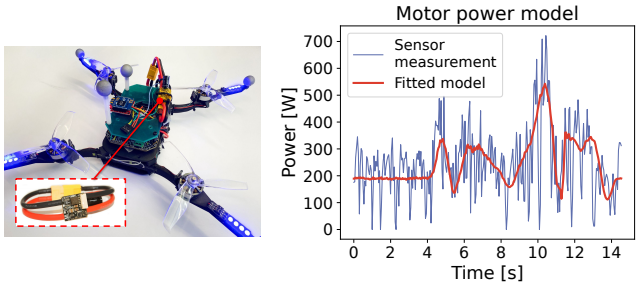
The low-fidelity method estimates energy consumption by applying a differential flatness transformation to sparsely sampled points along the piecewise polynomial trajectory. This transformation converts position, yaw, and their derivatives at each point into motor speeds based on the ideal quadrotor dynamics model. Energy consumption is estimated using the model from [5], which maps rotor speed ω_i to power P_i for each i-th motor.

$$F_i = \tau \omega_i^2 \tag{12}$$

$$P_i = \eta F_i^{1.5} \tag{13}$$

τ in (12), is a predefined coefficient relating the motor thrust F_i to a rotor angular speed. $\eta > 0$ in (13) is a coefficient that depends on air density and motor parameters [5]. A realistic value for coefficient η is determined from real-world flight

experiments, involving 50 random trajectories flown by our reference quadcopter, which is equipped with rotor speed sensors and an inline battery voltage and current sensor, as depicted in Figure 3a.



(a) (b)

Fig. 3: (a) The drone utilized for data collection, with the inline current and voltage sensor highlighted in the red box. (b) A real-world flight example using the reference quadcopter, illustrating the comparison between noisy sensor data and our fitted model.

The high-fidelity model uses the same motor speed to power mapping but with more accurate dynamics simulation and Lithium Polymer (LiPo) battery simulation. The motor speeds are obtained from the closed-loop dynamics simulation in [14], which includes a simplified spherical drag force. The LiPo battery simulation incorporates the model from [15], which estimates the open circuit voltage V_{OC} based on the state of charge (SOC), indicating the percentage of remaining battery capacity. Minimizing energy consumption is equivalent to reducing the SOC drop during a flight. As depicted in Figure 4a, the open circuit voltage decreases with battery usage, or as the SOC declines, which is a common characteristic of batteries. In addition to the nonlinear map of V_{OC}, our LiPo simulation accounts for the battery's internal resistance, R_{int}, a function of SOC and temperature, following the methodology outlined in [16]. Figure 4b illustrates how the battery's internal resistance varies with SOC at different temperatures. For our implementation, we assume a constant temperature of $23°C$. The output voltage V_{out} of the high-fidelity model is then given as follows:

$$V_{\text{out}} = V_{\text{OC}} - i \cdot R_{int}. \tag{14}$$

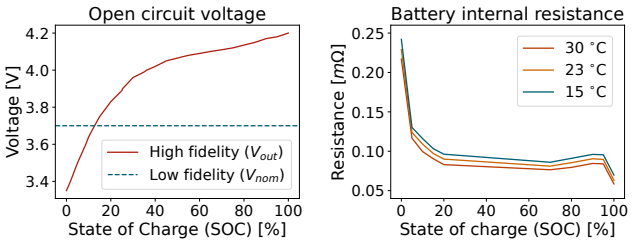
For the low-fidelity model, we assume that the output voltage is the nominal at $V_{\text{nom}} = 3.7V$ without simulating the battery.

The current draw is estimated from the output voltage; for high-fidelity models using V_{out}, and for low-fidelity models using V_{nom}. The battery on the reference drone is a 4-cell LiPo with a capacity of $Q_{\text{batt}} = 2\text{Ah}$, so the single cell voltage is multiplied by 4.

$$i_{\text{lf}} = \sum_{i=1}^{4} \frac{P_i}{4 \cdot V_{\text{nom}}}, \quad i_{\text{hf}} = \sum_{i=1}^{4} \frac{P_i}{4 \cdot V_{\text{out}}} \tag{15}$$

The final SOC for the trajectory is determined by integrating the current draw across the total flight duration.

$$\text{SOC}_{\text{hf}} = 100 \int_{t_0}^{t_f} i_{\text{hf}} / Q_{\text{batt}} \tag{16}$$



(a) (b)

Fig. 4: (a) The voltage model across different fidelity levels. (b) The internal resistance, which varies with temperature and state of charge (SOC). The ambient temperature for our simulations is set to 23°C.

$$\text{SOC}_{\text{lf}} = 100 \int_{t_0}^{t_f} i_{\text{lf}} / Q_{\text{batt}} \tag{17}$$

In the high-fidelity simulation, it is assumed that each trajectory begins with a fully charged battery at 100 % capacity, with the SOC representing the decrease in battery state over the flight.

B. Energy Prediction Model

We introduce an MFGP-based energy prediction model as a surrogate for the two forward-loop simulations discussed in the previous section. Integrating these simulations directly into trajectory optimization presents significant challenges due to the high computational demands and difficulty in optimization formulation. To address this, we develop an MFGP-based model capable of predicting the high-fidelity SOC (SOC_{hf}) for trajectories derived from sequences of waypoints and time allocations. This model serves as a reward estimator within the reinforcement learning framework for developing the planning policy. This approach adeptly manages the trade-off between accuracy and computational efficiency across various fidelity levels while enabling active adaptation of the prediction model during reinforcement learning.

The training data consists of waypoint sequences, where waypoint positions are randomly sampled in a unit cube $[-0.5, 0.5]^3$ and then scaled by the desired space scale L_{space}. The yaw component for each waypoint is set tangential to the local velocity of the minimum-snap trajectory, which is obtained by solving (4) without considering yaw values. Energy labels, defined as minimum-snap time allocations that minimize energy, are created through a two-step process: Initially, we perform nonlinear optimization as detailed in (4) to generate a normalized time allocation $\tilde{\mathbf{x}}_{\text{MS}}$. Following this, a line search procedure in (5) identifies a minimum-energy time allocation $\mathbf{x}_{\text{MS}}$. By scaling the time allocation with a grid of $\alpha \in [0.2, 2.0]$, we augment the dataset with more SOC simulations near the minimum energy time allocation. We generate a multi-fidelity dataset by deriving SOC estimates for the entire dataset using low-fidelity simulation, while estimating a smaller subset with high-fidelity simulation to reduce overall computation time.

The energy prediction model, illustrated in the center of Figure 2, predicts the SOC_{hf} of a trajectory obtained from a given waypoint sequence and time allocations. A recurrent

neural network (RNN) architecture with gated recurrent units (GRU) encodes the variable length input sequence into a latent embedding vector. The input waypoint sequence is a vector $\mathbf{s}^{\text{input}}_{\text{energy}}$ normalized by being divided by the dimension of the space and the time allocation by average trajectory time T_{avg}. The ith element is:

$$s^{\text{input}}_{\text{energy,i}} = [\tilde{p}^i_r/(L_{\text{space}}/2) \;\; \cos(\tilde{p}^i_\psi) \;\; \sin(\tilde{p}^i_\psi) \;\; f_{\text{EOS}} \;\; x_i/T_{\text{avg}}]. \tag{18}$$

f_{EOS} represents the end-of-sequence flag, being one at the last waypoints and zero elsewhere. T_{avg} is calculated by dividing the distance between waypoints by the average speed of the training trajectories. The latent embedding vector is obtained from the last hidden state of recurrent network through fully connected layers.

The multi-fidelity Gaussian process (MFGP) estimates energy consumption based on the embedding vector. The high-fidelity model's GP prior kernel leverages estimates from the low-fidelity kernel, as in (6). Energy consumption is typically linearly proportional to flight time. To capture more complex patterns beyond this simple relationship, our proposed model is trained to output normalized energy as follows:

$$y^{\text{label}} = (\frac{\text{SOC}}{\sum_i x_i} - \mu_{\text{SOC}})/\sigma_{\text{SOC}}. \tag{19}$$

$\sum_i x_i$ represents the total flight time of the trajectory. μ_{SOC} and σ_{SOC} represent the mean and standard deviation of the average SOC across all trajectories in the dataset. The output of the energy estimate function $\hat{f}_e$, is a Gaussian distribution defined by mean $\hat{\mu}_e$ and variance $\hat{\sigma}_e$. The energy prediction models are trained using the variational loss function in (10).

C. Planning Policy

By employing our multi-fidelity energy prediction model, we develop the planning policy via reinforcement learning. The policy is trained to output the minimum time allocation $\tilde{\mathbf{x}}$ given a variable-length waypoint sequence $\tilde{\mathbf{p}}$. The policy model's input is the normalized waypoint sequence, differing from the energy prediction model's input by excluding time allocation:

$$s^{\text{input}}_{\text{policy,i}} = [\tilde{p}^i_r/(L_{\text{space}}/2) \;\; \cos(\tilde{p}^i_\psi) \;\; \sin(\tilde{p}^i_\psi) \;\; f_{\text{EOS}}]. \tag{20}$$

Shown on the right side of Figure 2, it utilizes a model inspired by [17], replacing the energy prediction model's GP decoder with a GRU and an attention module for generating time allocation sequences. Additionally, a variational autoencoder (VAE) connects the encoder and decoder to prevent memorization. The decoder output at each recurrent step is transformed into the normalized time allocation x_i/T_{avg} using a multilayer perceptron (MLP).

The policy is initially pretrained to replicate the $\mathbf{x}_{\text{MS}}$ min-snap time allocations from the baseline method by minimizing the following loss function:

$$\mathcal{L}_{\text{Pretrain}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{MinSnap}}. \tag{21}$$

$\mathcal{L}_{\text{VAE}}$ is the evidence lower bound (ELBO) loss from the VAE, as in [18], and $\mathcal{L}_{\text{MinSnap}}$ is the mean squared error (MSE) loss between the output time allocations and the minimum snap time allocations. To further enhance training stability, the policy reproduces input waypoints $\tilde{\mathbf{p}}$ as outputs and minimizes the MSE between reconstructed and original waypoints ($\mathcal{L}_{\text{Recon}}$), alongside other losses.

The policy undergoes reinforcement learning training to optimize time allocations for minimal energy consumption. To improve convergence and mitigate training instabilities, we initialize the reinforcement learning process using a pretrained model. A reward signal of RL is designed to minimize the energy estimate while ensuring the trajectory remains feasible. To approximately enforce the feasibility, we apply a velocity constraint to the trajectory derived from the policy's time allocations. This involves setting a maximum speed, $v_{\max}$, derived from the equilibrium point at which the maximum thrust force and aerodynamic drag force are balanced. For the time allocation output by the policy, we generate the polynomial $\mathbf{p}(t)$ via (3) and find the norm of the maximum velocity $v_{\text{traj}} = \max_t(||\dot{\mathbf{p}}(t)||)$. A scaling factor $\phi \geq 1$ is applied to the time allocation to reduce the maximum velocity of the polynomial trajectory if $v_{\text{traj}} > v_{\max}$.

$$\phi = \max(1, \frac{v_{\text{traj}}}{v_{\max}}) \tag{22}$$

The minimum energy reward signal r_{energy} derived from the output of the energy prediction function $\hat{f}_e$ and ϕ as:

$$\hat{\mu}_e, \hat{\sigma}_e = \hat{f}_e(\phi\mathbf{x}, \tilde{\mathbf{p}}) \tag{23}$$

$$r_{\text{energy}}(\mathbf{x}, \tilde{\mathbf{p}}, \phi) = -\hat{\mu}_e - (\phi - 1). \tag{24}$$

The policy is updated with a clipping loss function from Proximal Policy Optimization (PPO) [19] called $\mathcal{L}_{\text{Policy}}$. The time allocation generation is modeled as a Markov Decision Process (MDP), where each time allocation x_i is an action that transitions the hidden state of the decoder RNN model.

The policy loss is combined with the waypoint reconstruction loss and VAE loss to get the RL loss:

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{Policy}}. \tag{25}$$

To enhance the accuracy of the energy prediction model further, we update the energy estimator $\hat{f}_e$ throughout the RL process. For each training batch, we select n_{hf} samples with the highest variance $\hat{\sigma}_e$ and estimate their actual energy labels, SOC_{hf}, using high-fidelity simulation. We also independently select n_{lf} samples with highest variance for low-fidelity labeling, SOC_{lf}. These simulation results are combined into the training dataset for the energy prediction model, which is updated at the end of each RL step with the new labels.

IV. Experiments

We present analysis to validate each component of our minimum energy planning algorithm. We specifically demonstrate the proposed method is efficient in the number of simulated data required, accurate in its energy estimate, and performant with respect to the minimum snap baseline method.

A. Data generation and model parameters

1) Energy simulation: The training dataset consists of randomly sampled waypoint sequences of lengths 5 to 14. Table I summarizes the number of generated sequences as well as the range of computation times. The testing dataset consists of 2×10^5 waypoint sequences generated in the same way. The waypoint sequences in the dataset are scaled to fit the target room dimensions of $L_{\text{space}} = [100\text{m}, 100\text{m}, 10\text{m}]$, simulating an outdoor environment suitable for longer flight times and more significant energy savings. The number of total samples generated for each fidelity and the approximate computation time is summarized in Table I. A single low-fidelity simulation requires a few milliseconds to complete, whereas a high-fidelity simulation takes a few seconds, each increasing with the number of waypoints and total path length. This results in a 1,000-fold increase in computation time from low- to high-fidelity evaluations.

	Number of training samples	Number of training samples	Simulation time per sample
Low-fidelity	2×10^6	1×10^5	$1{-}3.5$ms
High-fidelity	$100{-}1 \times 10^6$	1×10^5	$3{-}7$s

TABLE I: The size of the training data and the simulation time on each fidelity level.

The difference in energy prediction between the low- and high-fidelity model is 11% on average. In the following sections, we show that the multi-fidelity framework can incorporate the lower accuracy samples in a high-fidelity energy estimator.

2) Network parameters: The energy prediction encoder uses hidden layers of size 1024 for both GRUs and MLPs. The VAE module compresses the encoder's last hidden state into a 512-dimensional vector via a 4-layer MLP, which serves as the feature vector for the MFGP model. The policy network, designed for potential real-time mobile applications, has a smaller architecture. It uses hidden layers of size 256 for both encoder and decoder GRUs, as well as 3-layer MLPs with hidden size 256 for the fully connected portions.

B. Energy Estimator Sample Efficiency

We demonstrate the sample efficiency of the MFGP model by training it with different numbers of high-fidelity samples. SOC_{lf} and SOC_{hf} evaluations described in III-B are used to generate the multi-fidelity training data. Additionally, we compare the MFGP model with the single-fidelity model which is trained without the low-fidelity augmentation. This comparison with the single-fidelity model shows the value of including lower-fidelity data. The number of high-fidelity samples varies from 100 to 1×10^6, and the entire set of 2×10^6 low-fidelity samples are used on training the multi-fidelity model. Models trained with over 2,000 high-fidelity samples use 2,000 inducing points, while those with fewer than 2,000 samples use 500. Each model is trained for 100k iterations with batch size 200 with the Adam optimizer [20]

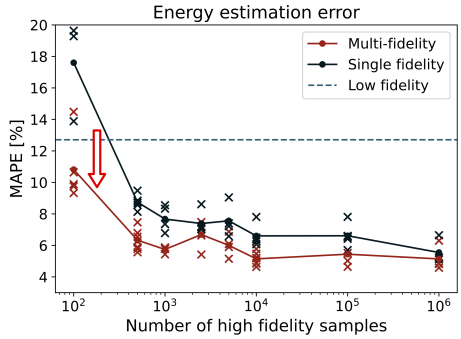


Fig. 5: Comparison of the mean absolute percentage error (MAPE) in SOC estimation when varying the number of high-fidelity samples. The multi-fidelity models are given approximately 2×10^6 low-fidelity evaluations as a second, lower-fidelity. The best MAPE from each model is marked with an "x", while the line indicates the average error over all samples. The dashed line is the MAPE from a single-fidelity model trained with only the low-fidelity samples.

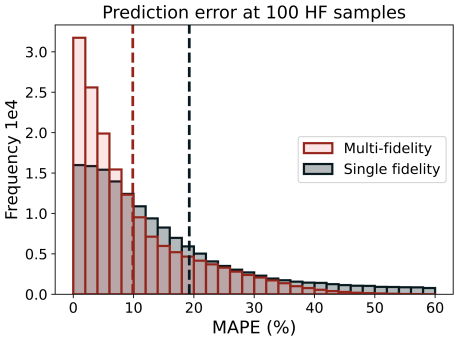


Fig. 6: These histograms compare the MAPE between a multi-fidelity model and a high-fidelity model, each given 100 high-fidelity (HF) samples. The dashed vertical lines indicate the mean error for each model.

and a learning rate of 1×10^{-4}. The models are compared via the testing dataset.

We compare the model performance using the mean absolute percentage error (MAPE), which for N samples, is defined as:

$$\mathcal{L}_{\text{MAPE}} = \frac{1}{N}\sum^{N}\left|\frac{\hat{f}_e(\mathbf{x}, \tilde{\mathbf{p}}) - \text{SOC}_{\text{hf}}}{\text{SOC}_{\text{hf}}}\right|. \tag{26}$$

As shown in Figure 5, both models achieve better mean absolute error as more high-fidelity data is supplied. As we decrease the amount of high-fidelity data, the MAPE of both models increases, but the multi-fidelity model maintains a significantly lower error. This implies that the multi-fidelity model achieves better high-fidelity sample efficiency. For the same error level, the single-fidelity model needs at least 10 times more data. For instance, to attain a 7% average error, the multi-fidelity framework only needs 1k HF samples, while the single-fidelity framework requires more than 10k. This performance difference underscores the potential importance of the multi-fidelity approach in applications with a higher evaluation cost.

Figure 6 illustrates a histogram of the MAPE for the single and multi-fidelity models trained with 100 high-fidelity (HF) samples. With such little data, both models report significant

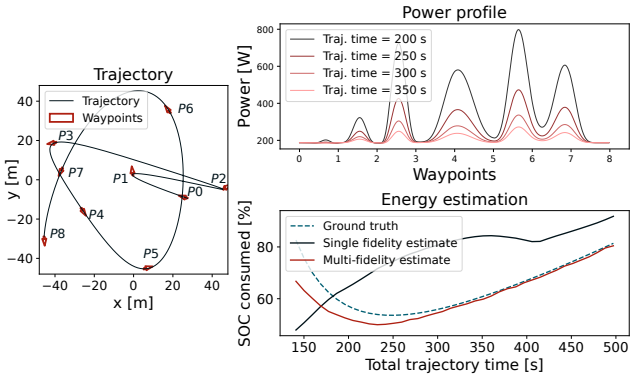


Fig. 7: Power profile and energy estimates for one 9-waypoint trajectory (left). Different power profiles and energy estimates are generated by linearly scaling a fixed time allocation to change the total trajectory time, and thus the total energy consumed. The energy estimates are done with the HF=100 models, where we see the single-fidelity model fail to estimate the energy curve correctly.

maximum error, but the multi-fidelity model performs much better, cutting the average error in half from about 20% to 10%.

An example trajectory energy estimate is shown in Figure 7. By linearly scaling a time allocation from the minimum snap baseline method, we can visualize the change in power usage and total energy consumption. The energy estimator should match the groundtruth SOC_{hf} calculated by (16). In this fixed time ratio example, the policy would ideally find the minimum energy time of about 250s. With only 100 high-fidelity samples, the single-fidelity GP model fails to correctly capture the single minimum energy point, and a policy optimization may not even converge. The low-fidelity samples still capture the simplest dynamics of the energy model, such as the asymptotes at very low and very high trajectory times. The multi-fidelity GP model incorporates this information and correctly captures the existence of a single minimum point, which would provide a better gradient to the policy optimization.

C. RL Policy Optimization

We evaluate the proposed RL framework by training a minimum energy planning policy with the energy prediction model. For the RL procedure, we select a pretrained reward model with 1k high-fidelity samples to showcase the sample efficiency of the method. The policy model is initialized through pretraining to replicate minimum snap time allocations $\mathbf{x}_{\text{ms}}$ with (21). From the variance estimate of the MFGP output, we re-label $n_{\text{lf}} = 16$ low-fidelity and $n_{\text{hf}} = 1$ high-fidelity sequences with the policy's time allocations to further train the reward estimator after each policy update. We run each of the RL experiments with a batch size of $N_{\text{batch}} = 600$, 50 batches per training step. The model is trained with the RMSProp optimizer [21] with a learning rate of 1×10^{-6}.

Table II contains the high level results comparing the minimum energy policy to the minimum snap baseline (*MinSnap*) labels. The minimum energy RL (*MinSnap MFGP*) improves on the baseline energy by 3.6% on average while reducing

	Energy reduction	Time reduction	Dynamics feasiblity	Energy feasiblity
MinSnap	0.0 %	0.0 %	100 %	97.6 %
MinTime	+60.1 %	-39.5 %	99.3 %	66.6 %
MinEnergy Sim	-4.67 %	-3.67 %	100 %	100 %
MinEnergy MFGP	**-3.59 %**	**-12.3 %**	**100 %**	**98.9 %**

TABLE II: Comparison of the planning policy trained with the proposed minimum energy (*MinEnergy MFGP*) framework, a minimum time objective (*MinTime*), and a minimum energy reward based only on the simulator (*MinEnergy Sim*). The policies are evaluated against the baseline model (*MinSnap*) in terms of energy improvement (*Energy reduction*) and time improvement (*Time reduction*). (*Dynamics feasibility*) and (*Energy feasibility*) indicate the percentage of trajectories with admissible motor commands and battery consumption, respectively.

the total time by 12.3%. We further compare our results with an RL policy trained to minimize flight time (*MinTime*) and a policy trained using exclusively simulated evaluations from SOC_{hf} (*MinEnergy Sim*). For the *MinTime* model, we train an additional model with a minimum time reward signal, defined as

$$r_{\text{time}}(\tilde{x}, \phi) = -\sum_i \phi \tilde{x}^i - (\phi - 1) \tag{27}$$

This reward signal minimizes time subject to the maximum velocity bound. The minimum time reward doesn't require an energy prediction component and can be trained without any simulated evaluation. Since total energy consumption is dependent on time, this model serves as another benchmark for evaluating our minimum energy policy.

The *MinTime* model completely disregards energy consumption to reduce the time by an average of 40% over the min-snap baseline. The cost of this approach is extremely high energy consumption—60% more than the baseline. The maximum velocity scaling effectively maintains dynamics feasibility for all three models, which means the motors are physically able to produce the rotor speeds required for the trajectories. However, the consequence of the high energy consumption is a drastic drop in energy feasibility, i.e., the proportion of trajectories that are flyable on a single battery charge. The minimum energy models shine in this metric with 98.9% of test trajectories within the policy's range, demonstrating the utility of explicitly optimizing over energy.

We further explore a case study to examine the effectiveness of the trained planning policy in reducing energy consumption. The trajectory illustrated in Figure 8 shows how the minimum energy policy closely follows the path of the minimum time trajectory, but it takes the sharp turn at waypoints 1 and 2 much slower. This drastically reduces the power spikes that the minimum time policy ignores. These 2.5kW peaks would strain the high performance battery in the real quadrotor.

Beyond energy savings, the RL policy reduces the computation time of the trajectory optimization. Traditional methods based on non-linear optimization require many iterations to determine time allocation ratios and total trajectory time. Conversely, our RL planning policy, on average, identifies more energy-efficient trajectories in a single neural network

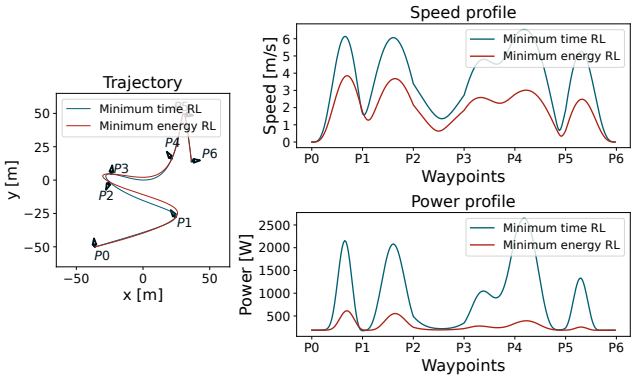


Fig. 8: Trajectory case study: For this 7-waypoint sequence (left), our minimum energy policy generates a time allocation that results in a 41.8% energy reduction and a 84.9% time increase over the minimum time solution.

inference, which takes only milliseconds to compute. Previous work on minimum time MFRL policies showed that this policy model can be applied to real-time trajectory planning on an embedded platform [6].

The multi-fidelity model also saves computation time during RL training. With 1000 training steps, fully simulating rewards to train the *MinEnergy Sim* model required 3×10^7 high-fidelity evaluations to achieve slightly better energy efficiency than *MinEnergy MFGP*. Instead, the proposed algorithm utilizes a proxy model and only evaluates the samples required to update the prediction model, thus drastically reducing the number of high-fidelity evaluations to only 5×10^4. This computational efficiency enables the sample-efficient training of a minimum energy policy that produces more energy-efficient trajectories with reduced execution time.

V. Conclusion

We introduce a multi-fidelity reinforcement learning approach for the efficient and accurate estimation of energy consumption in quadrotor trajectories. To mitigate the expense of acquiring high-fidelity, complex simulation data, we employ a multi-fidelity Gaussian process as the surrogate energy prediction model. This model integrates quick, low-fidelity evaluations to enhance the precision of high-fidelity estimates. Our multi-fidelity technique allows for the efficient training of the prediction model, achieving comparable accuracy with significantly fewer high-fidelity samples. The resultant energy prediction model is incorporated into an RL framework, providing a reward signal to a policy aimed at optimizing planning policy for maximum energy savings.

The experimental validation demonstrates a dramatic improvement in sample efficiency over single fidelity methods, reducing computational requirements for simulations. Future work could extend this approach to real-world environments, potentially decreasing physical tests needed in setups like battery state-of-charge measurement. Furthermore, our model is not constrained to quadrotors, suggesting that the same framework could be applied to various vehicle dynamics, including electric cars and urban air mobility vehicles, or even to different domains that require managing expensive evaluations.

References

[1] K. Karydis and V. Kumar, "Energetics in robotic flight at small scales," *Interface Focus*, vol. 7, no. 1, p. 20160088, 2017.

[2] N. Michel, Z. Kong, and X. Lin, "Energy-efficient UAV trajectory generation based on system-level modeling of multi-physical dynamics," in *2022 American Control Conference (ACC)*. IEEE, pp. 4119–4126.

[3] F. Yacef, N. Rizoug, L. Degaa, and M. Hamerlain, "Energy-efficiency path planning for quadrotor UAV under wind conditions," in *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, pp. 1133–1138.

[4] N. Kreciglowa, K. Karydis, and V. Kumar, "Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 656–662.

[5] H. Alkomy and J. Shan, "Investigating the effects of polynomial trajectories on energy consumption of quadrotors," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 3, pp. 1593–1604, 2023.

[6] G. Ryou, G. Wang, and S. Karaman, "Multi-fidelity reinforcement learning for time-optimal quadrotor re-planning," *arXiv preprint arXiv:2403.08152*, 2024.

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

[8] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González, "Deep gaussian processes for multi-fidelity modeling," *arXiv preprint arXiv:1903.07320*, 2019.

[9] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in neural information processing systems*, 2006, pp. 1257–1264.

[10] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13. Arlington, Virginia, USA: AUAI Press, 2013, p. 282–290.

[11] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification," in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 351–360.

[12] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration," in *Advances in Neural Information Processing Systems*, 2018.

[13] 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.

[14] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," *arXiv preprint arXiv:1905.11377*, 2019.

[15] G. Sethia, S. Majhi, S. K. Nayak, and S. Mitra, "Strict lyapunov super twisting observer design for state of charge prediction of lithium-ion batteries," vol. 15, no. 2, pp. 424–435.

[16] A. Aktas, Y. Kırcicek, and M. Ozkaymak, "Modeling and validation analysis according to temperature effect of different type batteries," vol. 24, no. 2, pp. 1031–1043.

[17] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," 2016.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *The International Conference on Learning Representations (ICLR)*, 2014.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization."

[21] P. Ghasemi, M. Karbasi, A. Zamani Nouri, M. Sarai Tabrizi, and H. M. Azamathulla, "Application of gaussian process regression to forecast multi-step ahead SPEI drought index," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5375–5392.

Acknowledgment

This work was supported in part by the Hyundai Motor Company.