Multi-fidelity Modeling and Reinforcement Learning for Energy Optimal Planning

by

Luke de Castro

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2024

Authored by: Luke de Castro
Department of Aeronautics and Astronautics
May 17, 2024

Certified by: Sertac Karaman
Professor of Aeronautics and Astronautics, Thesis Supervisor

Accepted by: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Multi-fidelity Modeling and Reinforcement Learning for Energy Optimal Planning

by

Luke de Castro

Submitted to the Department of Aeronautics and Astronautics
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

ABSTRACT

Modeling the energy consumption of a quadrotor involves complex electrical and physical dynamics, making it difficult to optimize over. We present a sequence-to-sequence multi-fidelity Gaussian process (MFGP) to learn a data-driven model to predict the energy required to fly a given vehicle trajectory. The goal is to create an accurate energy prediction that minimizes the number of expensive high fidelity simulations required for training. The MFGP algorithm can incorporate many low accuracy samples from a simple motor model with a few computationally demanding battery simulations to create a single accurate energy prediction. We perform sample efficiency experiments, finding 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 trajectory planner. The RL policy generates more energy efficient trajectories than those found by a nonlinear optimization baseline method, and we compare it to a minimum time RL model to show that the energy efficient policy is non-trivial.

Thesis supervisor: Sertac Karaman
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank my advisor Sertac Karaman for his generous support throughout my MIT career. His insightful guidance and vision helped me distill my work into coherent ideas and discoveries. I am honored to be his student and look forward to working with him in the future. This project would not have been possible without the help of Gilhyun Ryou. Not only did his work serve as a foundation for this project, his steadfast mentorship and collaboration saved me countless times during the research process. I would also like to thank the entire AERA lab for exposing me to fascinating research and lasting friendships. I enjoyed learning and working with them all. Finally, I thank my family for their unconditional love and support.

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

Contents

List of Figures

List of Tables

Chapter 1

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 1.1. This work aims to identify the balance between speed extremes for optimal energy efficiency.

While we apply the method to quadrotors, our multi-fidelity approach could be applied to other applications where the optimal decision depends on complex modeling. Optimizing over dynamics such as battery electrochemistry or vehicle aerodynamics requires evaluating a nonlinear model, numerical simulation, or real-world experiment. This work aims to minimize the number of model executions required to generate the optimal decision without compromising modeling power.

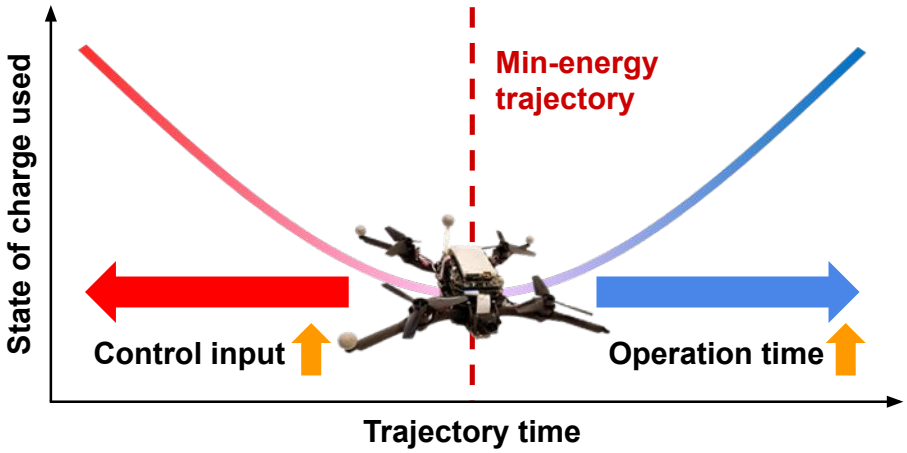


Figure 1.1: The main challenge in minimum energy trajectory planning involves balancing minimizing operating time with average power consumption. The proposed method provides an accurate power consumption model to precisely identify the optimal balance between these two objectives.

1.1 Motivation

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 system 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 energy-efficient planning methods 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 ap-

proaches often necessitate predetermining the final trajectory time, focusing solely on optimizing collocation points to reduce the complexity of the optimization process. Fixing the final time overlooks 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.

Deep learning has become a popular method to avoid the nonlinear optimization over time in trajectory generation [6]. Previous works have focused solely on time-optimal quadrotor planning [6], [7]. End-to-end reinforcement learning (RL), creating a policy mapping observation to control input, has achieved state-of-the-art performance in time optimal quadrotor control [8]. The trajectory planning policy trained in [7] finds trajectories that are faster than the minimum snap nonlinear optimization in only a couple milliseconds. We apply the latest advances in RL to minimum energy trajectory planning while incorporating high fidelity energy models.

1.2 Contribution

The 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 efficiently leveraging complex and realistic simulations to train the energy prediction model, avoiding the need to oversimplify battery models. We build on previous multi-fidelity techniques shown in [7], [9], [10], which model the dynamic feasibility boundary from data to guide RL training. Where these works define a classification problem, determining whether a given speed profile is feasible over a waypoint sequence, this work creates a regression problem to estimate the energy required to fly a trajectory.

- **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. Additionally, the accuracy of the energy prediction model is further improved through active learning, achieved by reevaluating policy outputs with high uncertainty estimates using actual simulations. This approach enables the trained model to generate trajectories that consume less energy on average compared to baseline minimum-snap trajectories previously considered to be highly energy-efficient.

Chapter 2

Preliminaries

2.1 Min-snap Trajectory Optimization

Minimum snap trajectory planning is a popular choice for quadrotors [11]. The method produces a fourth-order polynomial in position and a second order polynomial for yaw angle. These smooth polynomials can be generated quickly through convex optimization and facilitate trajectory tracking through a geometric controller, as in [12]. For a sequence of waypoints $\tilde{p}^i$, each consisting of a prescribed position $\tilde{p}^i_r \in \mathbb{R}^3$ and yaw angle $\tilde{p}^i_\psi \in \mathbb{S}^1$ ($\tilde{\mathbf{p}} = \begin{bmatrix} \tilde{p}^0 \dots \tilde{p}^m \end{bmatrix}$), 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:

$$\begin{aligned} &\underset{p=[p_r,p_\psi],\mathbf{x}\in\mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad \sigma\left(\mathbf{x}, p\right) \\ &\text{subject to} \;\; p(0) = \tilde{p}^0, \;\; p\left(\textstyle\sum_{j=1}^{i} x_j\right) = \tilde{p}^i, \; i = 1, \; \dots, \; m, \\ &\qquad\qquad p \in \mathcal{P} \end{aligned} \tag{2.1}$$

where

$$\sigma(\mathbf{x}, p) = \int_0^{\sum_{i=1}^{m} x_i} \left\| \frac{d^4 p_r}{d^4 t} \right\|^2 + \left(\frac{d^2 p_\psi}{d^2 t} \right)^2 dt. \tag{2.2}$$

$\mathcal{P}$ represents the set of feasible trajectories. 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:

$$\begin{aligned} \chi(\mathbf{x}, \tilde{\mathbf{p}}) = \underset{p=[p_r, p_\psi]}{\text{argmin}} \quad & \sigma\left(\mathbf{x}, p\right) \\ \text{subject to } \; p(0) = \tilde{p}^0, \;\; & p\left(\textstyle\sum_{j=1}^{i} x_j\right) = \tilde{p}^i, \; i = 1, \; \ldots, \; m, \end{aligned} \tag{2.3}$$

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, $\tilde{\mathbf{x}}$:

$$\underset{\tilde{\mathbf{x}} \in \mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad \sigma\left(\chi(\tilde{\mathbf{x}}, \tilde{\mathbf{p}}), \tilde{\mathbf{x}}\right) \;\; \text{subject to } \sum_{i=1}^{m} \tilde{x}_i = 1, \tag{2.4}$$

3) Line search to generate the feasible trajectory with the lowest energy, given an energy evaluation function f_e:

$$\underset{\alpha \in \mathbb{R}_{>0}}{\text{minimize}} \; f_e(\alpha \tilde{\mathbf{x}}, \tilde{\mathbf{p}}), \;\; \text{subject to } \; \chi(\alpha \tilde{\mathbf{x}}, \tilde{\mathbf{p}}) \in \mathcal{P}. \tag{2.5}$$

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 [11]. This method can be computationally expensive, depending on the f_e. We check for feasibility by constraining the commanded rotational speed of the motor. This optimization method serves as the baseline for comparing the energy consumption against our resulting trained policy.

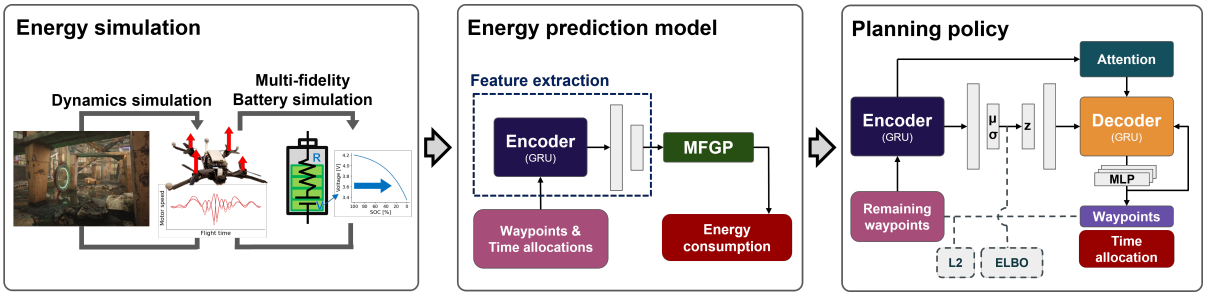


Figure 2.1: The three primary components of the minimum energy pipeline. First, we model the quadrotor energy dynamics in simulation. Then we train a multi-fidelity energy estimator with a sequence-to-sequence Gaussian process. Finally, we train another sequence-to-sequence model to generate the energy optimal time allocations.

2.2 Multi-Fidelity Gaussian Process

We employ the Multi-fidelity Gaussian Process (MFGP) to efficiently model the energy function based on sparse multi-fidelity evaluations. Gaussian processes (GP) are non-parametric Bayesian models that predict unseen data points from a collection of labeled data [13]. The GP assumes a finite set of data points has a joint Gaussian function defined by a mean function and covariance kernel. GPs are a popular choice to model complex physical systems such as combustion engine exhaust modeling [14] and weather forecasting [15]. The MFGP is a special form of GP, modeling the relationship between separate datasets that sample the same process at different fidelity levels.

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 [16]. An element of the l-th covariance kernel $k_l(z, z')$ is estimated as

$$\begin{aligned} k_l(z, z') = k_{l,corr}(z, z')(\sigma^2_{l,linear}\ f_{l-1}(z)^T f_{l-1}(z') \\ +k_{l,prev}(f_{l-1}(z), f_{l-1}(z'))) + k_{l,bias}(z, z'), \end{aligned} \tag{2.6}$$

where f_{l-1} is the Gaussian process estimation from the preceding fidelity level, $\sigma_{l,linear}$ is a con-

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

$$P(\mathbf{y}_l|\mathbf{Z}) = \mathcal{N}(\mathbf{y}_l|0, K_l(\mathbf{Z}, \mathbf{Z})), \tag{2.7}$$

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

$$P(y_*|\mathbf{z}_*, \mathbf{Z}_l, \mathbf{y}_l) = \int P(y_*|\mathbf{z}_*, \mathbf{Z}_l, \mathbf{y}_l) P(\mathbf{y}_l|\mathbf{Z}_l) d\mathbf{y}_l. \tag{2.8}$$

MFGP is further accelerated with the inducing points method [17]–[19], 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 $P(\mathbf{y}_l|\mathbf{Z}_l)$:

$$\mathcal{L}_{\mathcal{M}}(\mathbf{y}, \mathbf{Z}) = -\mathbb{E}_{q(\mathbf{u})}\left[\log P(\mathbf{y}|\mathbf{u})\right] + D_{\mathrm{KL}}\left[q(\mathbf{u})\|P(\mathbf{u}|\mathbf{Z})\right]. \tag{2.9}$$

$\mathcal{M} \coloneqq (\theta, \mathbf{m}, \mathbf{S})$ 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:

$$\mathcal{L}_{\mathrm{MFGP}} = \sum_{l=1}^{L} \mathcal{L}_{\mathcal{M}_l}(\mathbf{y}_l, \mathbf{Z}_l). \tag{2.10}$$

The MFGP is implemented with GPyTorch [20] based on the work presented in [9].

Chapter 3

Methods

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

$$\underset{\mathbf{x}}{\text{minimize}}\ f_e(\mathbf{x}, \tilde{\mathbf{p}}), \quad \text{subject to} \quad \chi(\mathbf{x}, \tilde{\mathbf{p}}) \in \mathcal{P}. \tag{3.1}$$

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 (3.1) 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.

3.1 Estimation of Energy Consumption

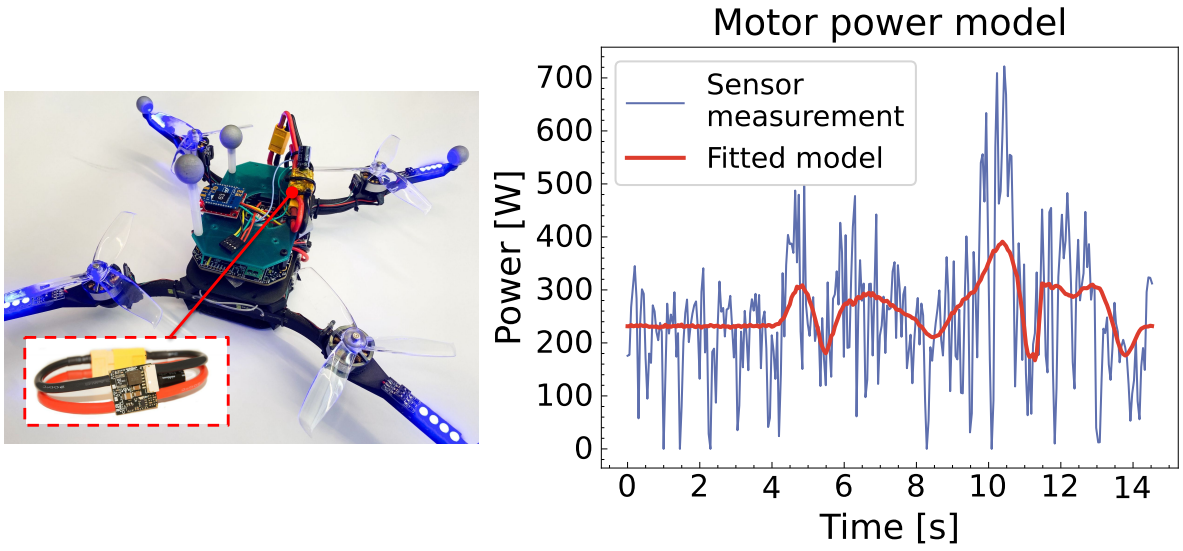
We use two methods, differing in accuracy and computational demand, to estimate energy consumption for planned trajectories. For given sets of time allocations and waypoint sequences, both models calculate the energy consumption of the piecewise polynomial trajectory derived from (2.3). Due to differences 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 and computational speed.

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.

$$P_i = \eta \omega_i^3 \tag{3.2}$$

$\eta > 0$ in (3.2) is a coefficient that depends on air density and motor parameters [5]. A realistic value for the coefficient η is determined from real-world flight experiments. Our reference quadcopter flies 50 random trajectories while equipped with rotor speed sensors and an inline battery voltage and current sensor, as depicted in Figure 3.1a. A linear regression model is then fit to create the motor power model.

The high-fidelity model uses the same motor speed to power mapping but with a more accurate dynamics simulation and Lithium Polymer (LiPo) battery simulation. The motor speeds are obtained from the closed-loop dynamics simulation in [21], which includes a simplified spherical drag force. The LiPo battery simulation incorporates the model from [22], 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



(a) (b)

Figure 3.1: (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.

flight. As depicted in Figure 3.2a, 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 [23]. Figure 3.2b 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:

$$V_{\text{out}} = V_{\text{OC}} - i \cdot R_{int}. \tag{3.3}$$

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.

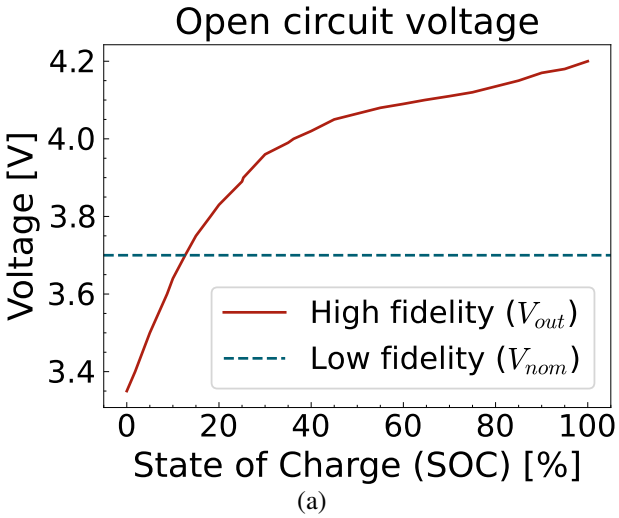
$$i_{\text{lf}} = \sum_{i=1}^{4} \frac{P_i}{4 \cdot V_{\text{nom}}}, \quad i_{\text{hf}} = \sum_{i=1}^{4} \frac{P_i}{4 \cdot V_{\text{out}}} \tag{3.4}$$

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

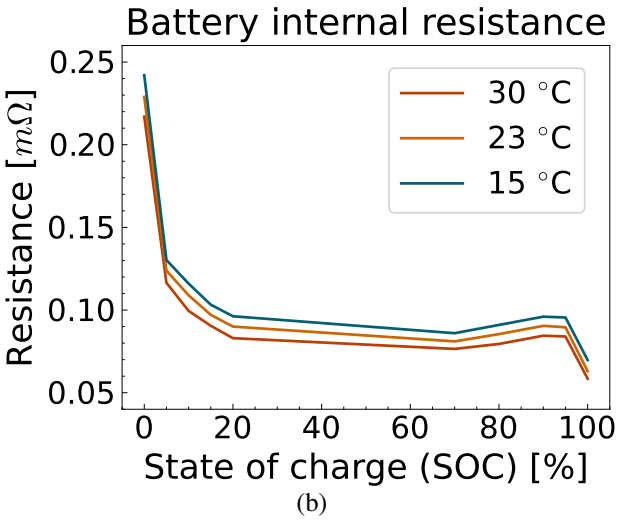
$$\text{SOC}_{\text{hf}} = 100 \int_{t_0}^{t_f} i_{\text{hf}} / Q_{\text{batt}} \tag{3.5}$$

$$\text{SOC}_{\text{lf}} = 100 \int_{t_0}^{t_f} i_{\text{lf}} / Q_{\text{batt}} \tag{3.6}$$

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.



(a)



(b)

Figure 3.2: (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.

3.2 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 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.

We first generate training data in a manner closely following the procedure from the time optimal trajectory planning work in [10]. Sequences of waypoints are randomly generated in a unit cube $[-0.5, 0.5]^3$. Samples are accepted based on two topological constraints: the summed Menger curvature [24] and the summed Euclidean distance between subsequent waypoints.

$$I_{\text{curvature}} = \sum_{i=0}^{m-2} \frac{1}{R(\tilde{p}_r^i, \tilde{p}_r^{i+1}, \tilde{p}_r^{i+2})} \in [5, 20] \tag{3.7}$$

$$I_{\text{distance}} = \sum_{i=0}^{m-1} d(\tilde{p}_r^i, \tilde{p}_r^{i+1}) \in [0, 30]. \tag{3.8}$$

An initial minimum-snap trajectory obtained from (2.4) is used to reject waypoint sequences that result in trajectories that go beyond $[-1, 1]^3$ and to set the yaw component for each waypoint tangential to the local velocity. The actual waypoint positions are obtained by scaling with the desired space scale L_{space}. Energy labels are created through a two-step process: initially, we perform nonlinear optimization as detailed in (2.4) to generate a normalized time allocation $\tilde{\mathbf{x}}_{\text{MS}}$. Following this, a line search procedure in (2.5) 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. SOC estimates for the entire dataset are derived using the low-fidelity simulation, while a smaller subset undergoes estimation with the high-fidelity simulation, in order to reduce the computation time.

The energy prediction model, illustrated in the center of Figure 2.1, predicts the SOC_{hf} of a trajectory obtained from a given waypoint sequence and time allocations. A sequence-to-sequence 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 normalized by being divided by the dimension of the space and the time allocation by average trajectory time

T_{avg}:

$$s^{\text{input}}_{\text{energy,i}} = [\tilde{p}^i_r/(L_{\text{space}}/2) \quad \cos(\tilde{p}^i_\psi) \quad \sin(\tilde{p}^i_\psi) \quad f_{\text{EOS}} \quad x_i/T_{\text{avg}}]. \tag{3.9}$$

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 (2.6). 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:

$$y^{\text{label}} = (\frac{\text{SOC}}{\sum_i x_i} - \mu_{\text{SOC}})/\sigma_{\text{SOC}}. \tag{3.10}$$

$\sum_i x_i$ 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 energy estimate, $\hat{f}_e$, is derived by denormalizing the GP output. The energy prediction models are trained using the variational loss function in (2.10).

3.3 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 the time allocations:

$$s^{\text{input}}_{\text{policy,i}} = [\tilde{p}^i_r/(L_{\text{space}}/2) \quad \cos(\tilde{p}^i_\psi) \quad \sin(\tilde{p}^i_\psi) \quad f_{\text{EOS}}]. \tag{3.11}$$

Shown on the right side of Figure 2.1, it utilizes a sequence-to-sequence model inspired by [25], 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:

$$\mathcal{L}_{\text{Pretrain}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{MinSnap}}. \tag{3.12}$$

$\mathcal{L}_{\text{VAE}}$ is the evidence lower bound (ELBO) loss from the VAE, as in [26], 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. The RL reward signal is designed to minimize the energy estimate while ensuring the trajectory remains feasible. The time optimal work presented in [9] discusses the feasibility boundary in depth, creating a multi-fidelity classification model to predict feasibility. To streamline our approach to minimizing energy, we apply a velocity constraint to the minimum snap trajectory derived from the policy's time allocations. Minimum-energy trajectories generally exhibit slower speeds compared to time-optimal trajectories and typically fall within the feasibility boundary. 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 (2.3) 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 $v_{\text{traj}} > v_{\max}$.

$$\phi = \max(1, \frac{v_{\text{traj}}}{v_{\max}}) \tag{3.13}$$

The minimum energy reward signal r_{energy} is then calculated based on the energy prediction function $\hat{f}_e$ and ϕ as:

$$r_{\text{energy}}(\mathbf{x}, \tilde{\mathbf{p}}, \phi) = -\hat{f}_e(\phi\mathbf{x}, \tilde{\mathbf{p}}) - (\phi - 1). \tag{3.14}$$

The policy is updated with a Proximal Policy Optimization (PPO) [27] clipped loss function. Policy outputs are 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 reward signal is only given at the end of sequence, so each reward is discounted such that for a sequence of length m, $r_i = \gamma^{m-i} r_{\text{energy}} \;\; (i = 1, 2, \ldots, m)$. The actions $x_i' \sim \pi_{\text{old}}(\cdot|x_i)$ are sampled from the policy output distribution of the previous training step. The policy loss, $\mathcal{L}_{\text{Policy}}$, is computed as:

$$\begin{aligned} r_{\text{clip},i} &= \min\left(r_{\pi,i} r_i, \text{clip}(r_{\pi,i}, 1-\epsilon, 1+\epsilon) r_i\right), \\ r_{\text{entropy},i} &= -\log \pi(x_i'|x_i), \\ \mathcal{L}_{\text{Policy}} &= \sum\nolimits_{i=k+1}^{m} r_{\text{clip},i} + r_{\text{entropy},i}, \end{aligned} \tag{3.15}$$

$r_{\pi,i} = \pi(x_i'|x_i)/\pi_{old}(x_i'|x_i)$ is the probability ratio between the new policy and old policy, and ϵ denotes the clipping range. The policy loss is minimized with the waypoint reconstruction loss and VAE loss:

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{Policy}}. \tag{3.16}$$

To enhance the accuracy of the energy prediction model further, we update the energy estimator $\hat{f}_e$ throughout the RL process. A subset of policy outputs, identified by the highest variance in energy prediction from the GP model, is selected for reevaluation using the energy simulation. These simulation results are then incorporated into the training dataset for the energy prediction model, which is updated at the end of each RL training epoch.

Chapter 4

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. In 4.2, we investigate the estimation accuracy of our energy prediction model with different amounts of high fidelity training data. In 4.3, we analyze the minimum energy planning policy obtained from RL training.

4.1 Data generation and model parameters

4.1.1 Energy simulation

The training data used for the analysis consists of 2×10^6 randomly sampled waypoint sequences of length 5 to 14. A subset of the training sequences are measured with the high-fidelity simulation. The test dataset includes 2×10^5 randomly sampled waypoint sequences. 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 4.1. 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. The parameters of the power model for our experimental setup are defined in Appendix A. The energy consumption simulations for the dataset were run through a CPU cluster with 28 compute nodes, totaling about 1,000 cores.

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

Table 4.1: 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 still incorporate the lower accuracy samples in a high fidelity energy estimator.

4.1.2 Network parameters

For the energy prediction encoder, the hidden layers of the GRUs and MLPs have a size of 1024, and the VAE module compresses the last hidden state of the encoder into a 512-dimensional vector via a 4-layer MLP. This serves as the feature vector for the MFGP model. The policy encoder-decoder network is smaller to accommodate potential real-time applications and memory constraints of a mobile platform. For this network the hidden layer and embedded layer sizes are all of size 256 with 3-layer MLPs for the fully connected portions. The decoder network to reconstruct the input to optimize over $\mathcal{L}_{\text{Recon}}$ in (3.12). All the neural network experiments were run on an Nvidia RTX 6000 Ada GPU.

4.2 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 3.2 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 [28] and a learning rate of 0.0001. The models are compared via a test dataset.

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

$$\mathcal{L}_{\text{MAPE}} = \frac{1}{N} \sum^{N} \left| \frac{\hat{f}_e(\mathbf{x}, \tilde{\mathbf{p}}) - \text{SOC}_{\text{hf}}}{\text{SOC}_{\text{hf}}} \right|. \tag{4.1}$$

As shown in Figure 4.1, 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 4.2 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 maximum

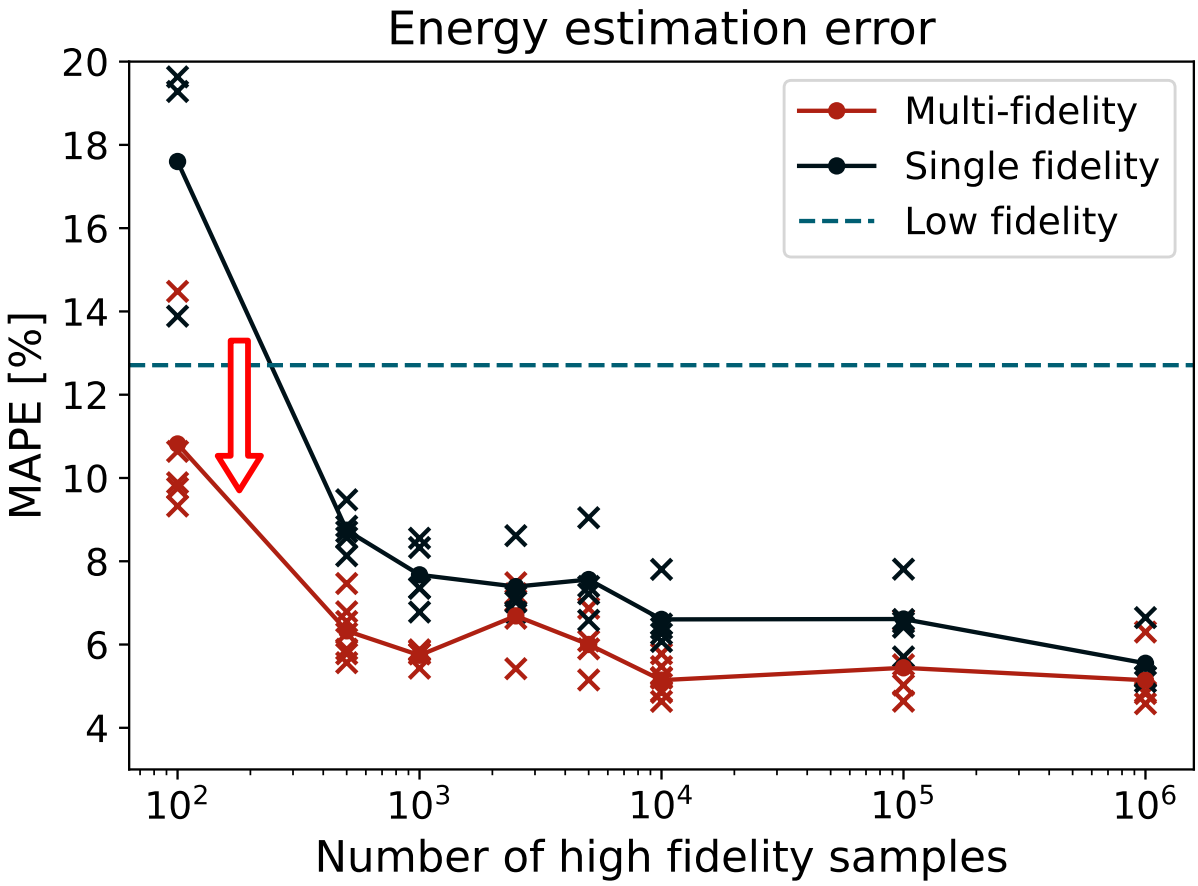


Figure 4.1: Comparison of the mean absolute percentage error (MAPE) in SOC estimation when varying the number of high fidelity samples. The multi-fidelity models are given approximately 2×10^6 low fidelity evaluations as a second, lower fidelity. The best MAPE from each model is marked with an "x", while 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.

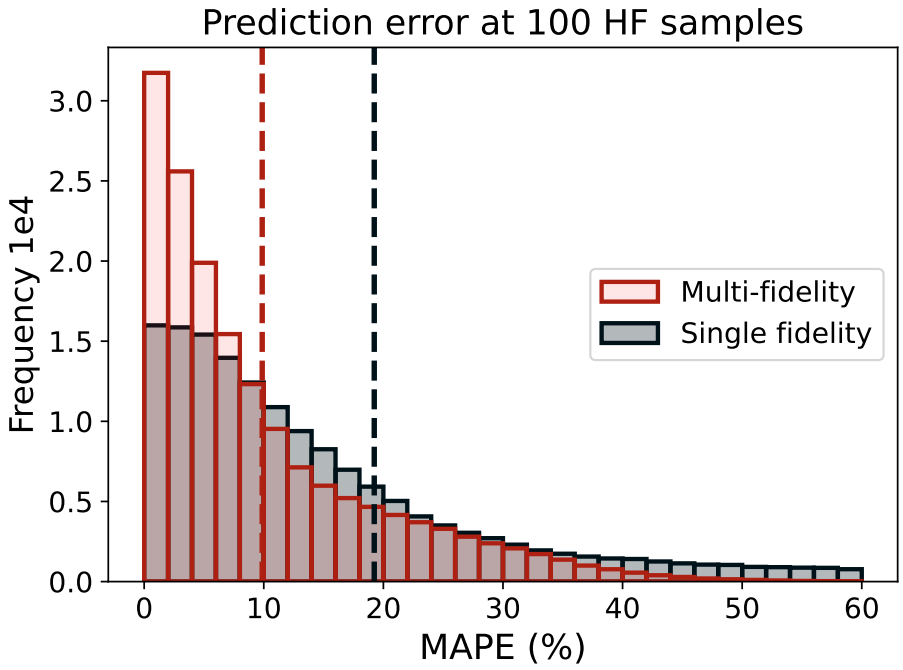


Figure 4.2: These histograms compare the MAPE between a multi-fidelity model and a high fidelity model, each given 100 high fidelity (HF) samples. The dashed vertical lines indicate the mean error for each model.

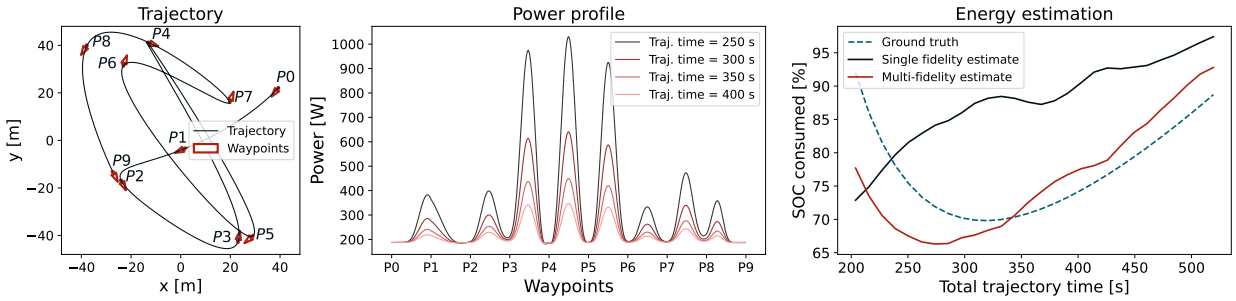


Figure 4.3: Power profile and energy estimates for one 10-waypoint trajectory (left). Different power profiles and energy estimates are generated by linearly scaling a fixed time allocation to change the total trajectory time, and thus the total energy consumed. The energy estimates are done with the HF=100 models, where we see the single fidelity model fail to estimate the energy curve correctly.

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 4.3. By linearly scaling a time allocation from the minimum snap baseline method, we can visualize the change in power usage and total energy consumption. With only 100 high fidelity samples, the single fidelity GP model fails to correctly capture the single energy minimum, which could lead the policy optimization to the wrong time allocation. 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.

4.3 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 (3.12). During RL, an active learning mechanism further refines the energy prediction model in addition to the policy update. From the variance estimate from the MFGP output, the algorithm selects 16 low fidelity and 1 high fidelity trajectories to further train the reward estimator. We run each of the RL experiments with a batch size of $N_{\text{batch}} = 200$, 50 batches per training step, and a clip parameter of $\epsilon = 0.01$ from (3.15). The model is trained with the RMSProp optimizer [15] with a learning rate of 1×10^{-6}. The maximum feasible speed, derived in Appendix A, is conservatively set to $v_{\max} = 6.9\text{m/s}$ to fully define the velocity rescaling factor ϕ in (3.14).

Since total energy consumption is dependent on time, another model is trained with a minimum time objective, defined as

$$r_{\text{mintime}}(\tilde{x}, \phi) = -\sum_i \phi \tilde{x}^i - (\phi - 1) \tag{4.2}$$

	Average energy reduction	Average time reduction	Percent dynamically feasible	Percent in feasible range
Minimum snap baseline	0.0 %	0.0 %	100 %	97.6 %
Minimum energy RL	-3.59 %	-12.3 %	100 %	98.9 %
Minimum time RL	+60.1 %	-39.5 %	99.3 %	66.6 %

Table 4.2: Policy optimization results starting with a minimum energy and a minimum time objective. We compare the two policies to the minimum snap baseline model through both energy improvement and time improvement. A percentage indicates an energy or time saving over the minimum snap baseline. The ratio of dynamically feasible models and trajectories within the feasible battery range is also listed.

which minimizes time subject to the maximum velocity bound. It does not require the energy prediction model and can serve as another benchmark for evaluating the minimum energy policy.

Table 4.2 contains the high level results comparing the minimum energy and minimum time RL models to the minimum snap baseline labels. The minimum energy RL improves on the baseline energy by 3.6% on average while reducing the total time by 12.3%. The model is able to effectively trade off between operating time reduction and control input increases.

The minimum time RL 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 dynamic 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 the drastic drop in the proportion of trajectories that are flyable on a single battery charge. The minimum energy RL model shines in this metric with 98.9% of test trajectories within the policy's range, demonstrating the utility of explicitly optimizing over energy.

Figures 4.4 and 4.5 compare the energy optimal RL policy to the min-snap baseline and time optimal RL policy respectively. Comparing to the baseline in Figure 4.4, the minimum energy policy does not always improve upon the energy consumption, which is compensated by the long tail skewing the distribution toward a more energy efficient mean. It flies faster than the baseline for all test trajectories, showing that total operating time is an important optimization variable for minimizing energy. The difference is greater when comparing the energy optimal and time optimal RL policies in Figure 4.5. In terms of energy efficiency, the energy optimal RL is a drastic

improvement over a pure time optimal approach, saving up to 55% total energy.

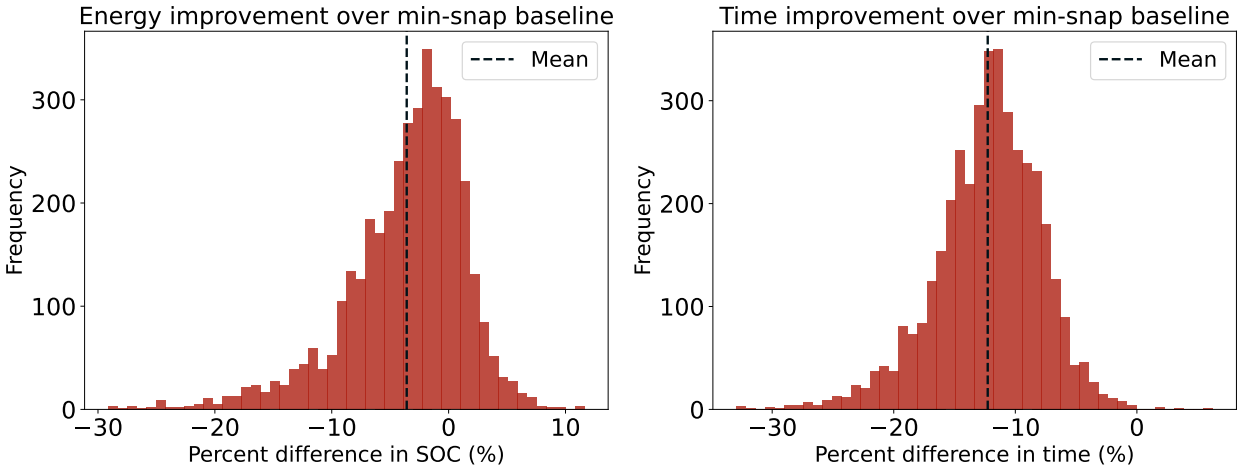


Figure 4.4: Comparison between the minimum energy RL policy and the min-snap baseline policy.

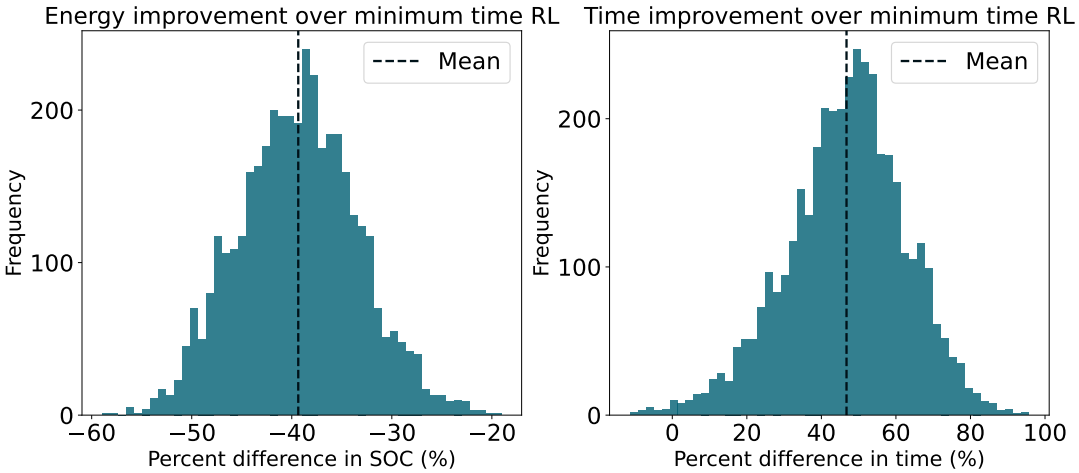


Figure 4.5: Comparison between the minimum energy RL policy and the minimum time baseline policy.

We further explore a case study to examine the effectiveness of the trained planning policy in reducing energy consumption. The trajectory illustrated in Figure 4.6 shows how the minimum energy policy closely follows that 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 3kW peaks would strain the high performance battery in the real quadrotor.

Beyond energy savings, it's important to highlight the RL policy's advantage in reducing trajectory optimization computation time. Traditional methods based on non-linear optimization require

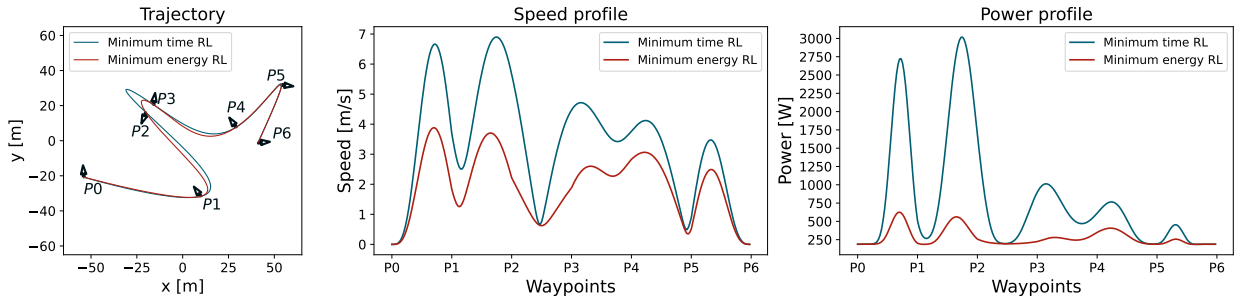


Figure 4.6: Trajectory case study. For this 7 waypoint sequence (left), our minimum energy policy generates a time allocation that results in a 44.6% energy reduction and a 50.5% time increase over the minimum time solution.

several minutes to determine time allocation ratios and total trajectory time. Conversely, our RL planning policy, on average, identifies more energy-efficient trajectories within milliseconds, significantly enhancing computational efficiency. Previous work on minimum time MFRL policies showed that this policy model can be applied to real-time trajectory planning on an embedded platform [7].

The active learning scheme also saves computation time during the RL training. With 1000 training steps, 50 batches per step and 600 trajectories per batch, simulating the reward for each sample would require 30 million high fidelity evaluations. With the active learning scheme only selecting a single trajectory to simulate in each batch, the number of high fidelity evaluations drops drastically to only 50 thousand. The minimum energy policy is sample efficient in training, quick to execute, and produces more energy efficient trajectories than the other policies.

Chapter 5

Conclusion

We introduce a multi-fidelity reinforcement learning approach for the efficient and accurate estimation of energy consumption in quadrotor trajectories. To mitigate the computational 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.

Our work opens multiple avenues for improvement and further research. Leveraging low-fidelity estimates to minimize the need for high-fidelity evaluations facilitates the incorporation of more complex evaluations, which are typically costlier. For example, more complex simulations or real-world flight experiments could be incorporated as additional fidelity levels. Furthermore, our model is not constrained by simulation environments, 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.

Appendix A

Quadrotor Parameters

Physical parameter	**Value**
Thrust coefficient τ	$1.6201 \times 10^{-6}\mathrm{N/rpm^2}$
Power coefficient η	$1.625 \times 10^{-3}\mathrm{W/rpm^3}$
Mass m	$0.771\mathrm{kg}$
Drag C_d	$1.\mathrm{kg/m}$
Maximum rotor speed $\omega_{\max}$	2750 rpm
Maximum feasible speed $v_{\max}$	6.9 m/s

Table A.1: Relevant physical parameters for the quadrotor model.

Table A.1 shows the values used to simulate the quadrotor. The thrust coefficient, mass, and maximum rotor speed are taken from the default values in the Flightgoggles dynamics simulator [21]. The drag coefficient is increased from 0.1 to 1.0 to ensure that the minimum time trajectory is sufficiently different from the minimum energy trajectory. The maximum feasible speed $v_{\max}$ is then derived from these parameters as follows:

We assume the motors spin at the maximum feasible rate to find the maximum total thrust force from all 4 motors $F_{T,\max}$.

$$F_{T,\max} = 4\tau\omega_{\max}^2 \tag{A.1}$$

Assuming the vehicle is flying at a constant velocity, the attitude θ is fully determined by

$$F_{T,\max} \sin\theta = mg \tag{A.2}$$

$$F_{T,\max} \cos\theta = F_d = C_d ||v_{\max}||^2 \tag{A.3}$$

F_d is the quadratic drag force parameterized by C_d. Solving the force balance, we arrive at $v_{\max} = 6.9\mathrm{m/s}$, which serves as the conservative upper bound on velocity in the optimization.

References

[1] K. Karydis and V. Kumar, "Energetics in robotic flight at small scales," *Interface Focus*, vol. 7, no. 1, p. 20 160 088, Feb. 6, 2017, ISSN: 2042-8898, 2042-8901. DOI: 10.1098/rsfs.2016.0088. URL: https://royalsocietypublishing.org/doi/10.1098/rsfs.2016.0088 (visited on 06/12/2023).

[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)*, Atlanta, GA, USA: IEEE, Jun. 8, 2022, pp. 4119–4126, ISBN: 978-1-66545-196-3. DOI: 10.23919/ACC53348.2022.9867646. URL: https://ieeexplore.ieee.org/document/9867646/ (visited on 01/17/2024).

[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)*, Prague, Czech Republic: IEEE, Jun. 29, 2020, pp. 1133–1138, ISBN: 978-1-72815-953-9. DOI: 10.1109/CoDIT49905.2020.9263968. URL: https://ieeexplore.ieee.org/document/9263968/ (visited on 01/16/2024).

[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)*, Miami, FL, USA: IEEE, Jun. 2017, pp. 656–662, ISBN: 978-1-5090-4495-5. DOI: 10.1109/ICUAS.2017.7991496. URL: http://ieeexplore.ieee.org/document/7991496/ (visited on 06/09/2023).

[5] H. Alkomy and J. Shan, "Investigating the effects of polynomial trajectories on energy consumption of quadrotors," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2022, ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2022.3220086. URL: https://ieeexplore.ieee.org/document/9954927/ (visited on 06/09/2023).

[6] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar, "Deep learning for optimization of trajectories for quadrotors," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2479–2486, Mar. 2024, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2024.3357399. URL: https://ieeexplore.ieee.org/document/10412114/ (visited on 02/16/2024).

[7] G. Ryou, G. Wang, and S. Karaman, *Multi-fidelity reinforcement learning for time-optimal quadrotor re-planning*, Mar. 12, 2024. arXiv: 2403.08152[cs]. URL: http://arxiv.org/abs/2403.08152 (visited on 05/15/2024).

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 31, 2023, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-023-06419-4. URL: https://www.nature.com/articles/s41586-023-06419-4 (visited on 05/15/2024).

[9] 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, pp. 1352–1369, Dec. 2021, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/02783649211033317. URL: http://journals.sagepub.com/doi/10.1177/02783649211033317 (visited on 12/14/2022).

[10] G. Ryou, E. Tal, and S. Karaman, "Real-time generation of time-optimal quadrotor trajectories with semi-supervised seq2seq learning," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, Dec. 14, 2023, pp. 1860–1870. URL: https://proceedings.mlr.press/v205/ryou23a.html.

[11] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China: IEEE, May 2011, pp. 2520–2525, ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980409. URL: http://ieeexplore.ieee.org/document/5980409/ (visited on 12/14/2022).

[12] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 4282–4288.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning* (Adaptive computation and machine learning). Cambridge, Mass: MIT Press, 2006, 248 pp., OCLC: ocm61285753, ISBN: 978-0-262-18253-9.

[14] X. Shi, D. Jiang, W. Qian, and Y. Liang, "Application of the gaussian process regression method based on a combined kernel function in engine performance prediction," *ACS Omega*, vol. 7, no. 45, pp. 41 732–41 743, Nov. 15, 2022, ISSN: 2470-1343, 2470-1343. DOI: 10.1021/acsomega.2c05952. URL: https://pubs.acs.org/doi/10.1021/acsomega.2c05952 (visited on 05/15/2024).

[15] 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, Dec. 2021, ISSN: 11100168. DOI: 10.1016/j.aej.2021.04.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S1110016821002647 (visited on 05/15/2024).

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

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

[18] 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, Bellevue, WA: AUAI Press, 2013, pp. 282–290.

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

[20] 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.

[21] 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.

[22] 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," *IET Renewable Power Generation*, vol. 15, no. 2, pp. 424–435, Feb. 2021, ISSN: 1752-1416, 1752-1424. DOI: 10.1049/rpg2.12039. URL: https://onlinelibrary.wiley.com/doi/10.1049/rpg2.12039 (visited on 01/24/2024).

[23] A. Aktas, Y. Kırcicek, and M. Ozkaymak, "Modeling and validation analysis according to temperature effect of different type batteries," *Thermal Science*, vol. 24, no. 2, pp. 1031–1043, 2020, ISSN: 0354-9836, 2334-7163. DOI: 10.2298/TSCI190401331A. URL: https://doiserbia.nb.rs/Article.aspx?ID=0354-98361900331A (visited on 01/18/2024).

[24] K. Menger, "Untersuchungen über allgemeine metrik. vierte untersuchung. zur metrik der kurven," *Mathematische Annalen*, vol. 103, pp. 466–501, 1930.

[25] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, *Generating sentences from a continuous space*, 2016. arXiv: 1511.06349 [cs.LG].

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

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

[28] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, Jan. 29, 2017. arXiv: 1412.6980[cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 05/18/2024).