Sequence Modeling for Time-Optimal Quadrotor Trajectory Optimization with Sampling-based Robustness Analysis

Katherine Mao[1], **Hongzhan Yu**[2], **Ruipeng Zhang**[2], **Igor Spasojevic**[1], **M. Ani Hsieh**[1], **Sicun Gao**[2], **Vijay Kumar**[1]
[1]University of Pennsylvania, [2]University of California San Diego
{maokat, igorspas, mya, kumar}@seas.upenn.edu
{ruz019, hoy021, sicung}@ucsd.edu

Abstract: Time-optimal trajectories drive quadrotors to their dynamic limits, but computing such trajectories involves solving non-convex problems via iterative nonlinear optimization, making them prohibitively costly for real-time applications. In this work, we investigate learning-based models that imitate an model-based time-optimal trajectory planner to accelerate trajectory generation. Given a dataset of collision-free geometric paths, we show that learning-based approaches can effectively learn the patterns underlying time-optimal trajectories. We introduce a quantitative framework to analyze local analytic properties of the learned models, and link them to the Backward Reachable Tube of the geometric tracking controller. To enhance robustness, we propose a data augmentation scheme that applies random perturbations to the input paths at training. Compared to classical planners, our method achieves substantial speedups, and we validate its real-time feasibility on a hardware quadrotor platform. Experiments demonstrate that the learned models generalize to previously unseen path lengths. The code for our approach can be found here: https://github.com/maokat12/lbTOPPQuad

Keywords: Trajectory Planning, Imitation Learning, Robustness Analysis, Aerial Robotics

1 Introduction

Optimal trajectory generation is one core component of an agile micro aerial vehicle's (MAVs) autonomy stack. Numerous applications such as search and rescue operations, disaster response, and package and aid delivery require these robots to perform tasks safely, at operational speeds. The aim of minimizing task completion time arises not only due to the limited battery life onboard the MAVs, but also the desire to take advantage of their full flight envelope.

The key algorithmic challenge behind optimization problems underlying synthesizing time-optimal trajectories lies in the non-convexity. One source of non-convex constraints comes from the presence of obstacles in the environment. Another arises from the nonlinear nature of the robot's dynamics. The majority of previous approaches optimize trajectories using a combination of simplified dynamics models, and/or faithful dynamics models with proxy actuation constraints. Other approaches plan trajectories with both faithful dynamics models and suitable actuation constraints. Such planners exhibit superior mission execution time, at the cost of far greater computational resources.

This is the first work to develop a learning-based algorithm for computationally efficient time optimal path parametrization for quadrotors with faithful dynamics and actuation constraints. The approach for learning a solution to the high-dimensional sequential optimization problem rests on a combination of domain-specific insights as well as a novel imitation learning formulation for "sequence-to-sequence" problems. In summary, the contributions of this paper are as follows:

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

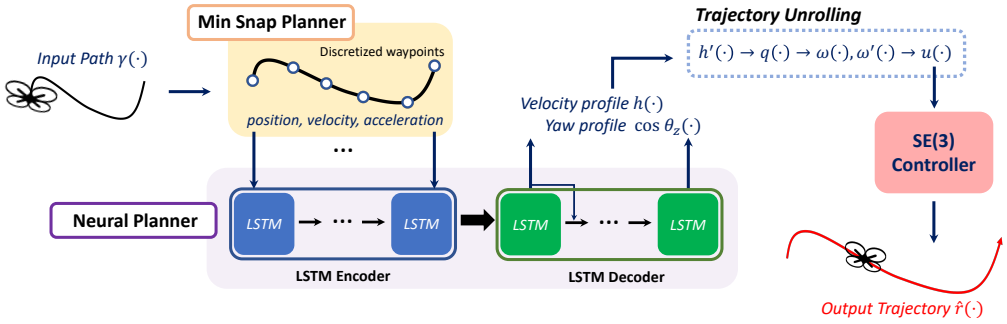


Figure 1: Overall pipeline. We begin by discretizing the input geometric path $\gamma(\cdot)$ into equally spaced grid points using a minimum-snap planner. Based on this discretized trajectory, our LSTM encoder-decoder model (which performed the best in our ablation study) predicts squared-speed $h(\cdot)$ and yaw θ_z profiles, imitating the model-based time-optimal planner TOPPQuad [1]. From these predicted profiles, we unroll the full robot state trajectory. Finally, a low-level geometric controller computes the control trajectory to execute, ensuring that per-motor actuation constraints must be satisfied.

- A learning-based formulation for imitating an model-based time-optimal trajectory planner, employing an input–output feature design that predicts only the minimal variables necessary to reconstruct the time-optimal trajectory.
- A rigorous robustness analysis framework that quantifies how well predicted trajectories can be tracked by a controller via a sampling-based approach to recover Backward Reachable Tubes (BRT), along with a data augmentation strategy to enhance model robustness.
- A comprehensive ablation study across various neural architectures, showing that an LSTM encoder-decoder model achieves near-time-optimal performance with significant speedup over optimized-based planners.
- A demonstration of our learning-based planner on a hardware platform, showing robust generalization to unseen path geometries and lengths.

2 Related Work

The nature of time-optimal quadrotor trajectories requires plans that push the system to its physical limits. Paired with the quadrotor's underactuated dynamics, this remains a challenging problem. Popularized by [2], many approaches plan trajectories following a polynomial structure, where trajectories are characterized by a set of waypoints and their time allocations [3][4][5][6]. These approaches take advantage of the quadrotor's differential flatness property to ensure smoothness of state values and often allow for faster compute times, but sacrifice the 'bang-bang' behavior required for more aggressive flight. Other approaches forgo the polynomial structure to plan around the full dynamics. A series of approaches track progress along a nominal path, allowing deviations [7][8], while [1] determines the time paramterization of a fixed geometric path. Finally, [9], [10] simplify the quadrotor to a point model in the planning phase, then rely on modern nonlinear Motion Predictive Controllers (NMPC) to track a potentially dynamically infeasible reference trajectory.

The heavy computation cost of optimization-based time-optimal trajectory planners has made learning-based solutions an attractive option. Although polynomial planners for a set of waypoints can be formulated as a convex problem, the selection of the such waypoints and their corresponding time allocations is still computationally challenging. Approaches to the time allocation problem for a set of waypoints have used Gaussian Processes [11], sequence-to-sequence learning [12], GNNs [13] and transformers [14]. [15] broadens this problem and trains an LSTM to learn both the intermediary waypoints and time allocation given a series of safe flight corridors and goal points. While often computationally faster, the lack of full dynamics constraints in the polynomial approaches can lead to dynamically infeasible motor thrusts. On the other hand, planners that utilize the full dynamics can become intractably slow, ranging from seconds [1] to hours [7]. [16], [17] solve this by utilizing

Reinforcement Learning to generate near-time optimal trajectories for drone racing tracks. However, their progress-based nature encourages deviations from a nominal center-line path and cannot guarantee safety in cluttered environments and policies are overfit to known tracks.

3 Preliminary

TOPPQuad [1] is an optimization-based approach for Time-Optimal Path Parameterization (TOPP) which generates dynamically feasible quadrotor trajectories for a given collision-free path. The key to this method is the squared-speed profile, $h(\cdot)$, which dictates the relationship between traversal time and the progress along a N-point discretized geometric path $\gamma(\cdot)$. However, due to the non-convexity of the full dynamics model and the desire for explicit bounds on actuation constraints, the optimization must be performed upon an $16 \times N$ variable state space, incurring heavy computation cost and slow runtime. This state space includes the rotational profile along the trajectory. Conversely, the quadrotor is a differentially flat system [2], where any trajectory can be uniquely represented by four flat variables and their derivatives: position (x, y, z) and yaw (θ_z). Given the nature of the TOPP problem and the relationship between h and the higher positional derivatives, the time optimal trajectory along a given path $\gamma(\cdot)$ can be represented as function of just h and θ_z.

4 Methodolgy

4.1 Imitation Learning Problem Formulation

When designing an imitation learning framework, it is essential to ensure the input-output mapping is feasible with respect to system constraints. Restricting the output dimensionality to only the minimal set of required features mitigates overfitting risks and promotes robustness. Given $\gamma(\cdot)$, TOPPQuad produces a time-optimal, dynamically feasible trajectory $r(\cdot)$, where dynamically feasibility is defined as respecting all state and input constraints. However, the variables are tightly coupled by the underlying dynamic constraints, making it challenging to learn jointly in a direct manner. In particular, the motor thrust $u(\cdot)$ must lie within specified actuation limits, highly non-linear functions of the flat variables, and the quaternions $q(\cdot)$ must reside on the 3-sphere, $S^3 = \{q \in \mathbb{R}^4 \mid ||q||_2 = 1\}$.

We propose to use $[h(\cdot), \cos\theta_z(\cdot)]$ as the output variables, where $\cos\theta_z(\cdot)$ encodes the yaw rotation enconded via the cosine function in order to address the many-to-one yaw wraparound. Next, we discuss how to recover the original variables, providing equations in Appendix B. We obtain the speed profile derivative $h^{'}$ from finite differences of the learned squared-speed profile h. To construct quaternion q, we first compute the body z-axis vector b_3 by adding gravity to the derived acceleration (from speed profiles and path curvature) and normalizing, ensuring b_3 aligns with the net thrust vector. Then, q is derived via rotation composition, orienting the drone to align its body-z axis with b_3 and setting yaw to the desired θ_z. Next, we calculate the rotation change between steps, yielding angular velocity ω and its derivative ω'. Finally, a low-level geometric controller [18][19] computes $u(\cdot)$ from the derived states. In TOPPQuad, $u(\cdot)$ is jointly optimized with other decision variables to guarantee dynamic feasibility. Conversely, our approach recovers $u(\cdot)$ from the orientation and accelerations and the low-level controller applies a clipping function to ensure the per-motor actuation constraints are satisfied.

We augment the model's input with path curvature, the first and second derivatives of the geometric path $\gamma'(\cdot)$ and $\gamma''(\cdot)$. This directly provides explicit geometric information, rather than requiring the model to infer it implicitly. In summary, we formulate the imitation task as learning the mapping:

$$[\gamma(\cdot), \gamma'(\cdot), \gamma''(\cdot)] \rightarrow [h(\cdot), \cos\theta_z(\cdot)], \tag{1}$$

the minimal set of outputs necessary to reconstruct the time-optimal path parameterization.

4.2 Robustness Analysis

To ensure practical reliability, learned trajectory planners must be explicitly evaluated for dynamic infeasiblity, rather than relying solely on empirical performance metrics.

Given a geometric path $\gamma(\cdot)$, the model predicts $[h(\cdot), \cos\theta_z(\cdot)]$ from which we derive the robot state trajectories $r(\cdot) := \{h(\cdot), q(\cdot), \omega(\cdot)\}$. We assume that the quadrotor starts at $\gamma(s_1)$ at rest (zero velocity). Let $\hat{r}(\cdot)$ denote the full robot state trajectory after executing the error-tracking low-level controller U [19]. That is, $\hat{r}(s_1) = r(s_1)$, and for each $i \in \{1, ..., N-1\}$:

$$\hat{r}(s_{i+1}) = \hat{r}(s_i) + \int_0^{\Delta s} \dot{f}(\hat{r}(s_i + s), U\Big(\hat{r}(s_i + s), r(s_i)\Big))\mathrm{d}s, \tag{2}$$

which integrates the quadrotor dynamics $\dot{f}$ over the spatial step-size Δs. If the model's predictions are perfectly dynamically feasible, the planned and executed trajectories coincide, i.e., $r(\cdot) = \hat{r}(\cdot)$. Conversely, a state prediction is dynamically infeasible when the planned state is unreachable from the executed state under the system dynamics.

We quantify dynamic feasibility via *trackability* using the concept of finite-time Backward Reachable Tube (BRT) [20]: for state $x \in \mathcal{X}$ and time $\Delta t > 0$, let $\xi_U(x, \Delta t)$ denote the set of states from which x can be reached within Δt seconds under U:

$$\xi_U(x, \Delta t) = \{x_0 \in \mathcal{X} \mid \exists\, \tau \leq \Delta t, \text{ s.t. } x_0(\tau) = x \text{ under } U\}. \tag{3}$$

Proposition 4.1 *Suppose $\gamma(\cdot)$ is a geometric path. Let $r(\cdot)$ and $\hat{r}(\cdot)$ be the planned and the simulated trajectories, respectively. If, for each $i \in \{1, ..., N-1\}$,*

$$\hat{r}(s_i) \in \xi_U(r(s_{i+1}), t_i) \text{ where } t_i = 2\Delta s/(\sqrt{h(s_i)} + \sqrt{h(s_{i+1})}) \text{ (Eq. 22 in [1])}, \tag{4}$$

then the trajectory planner is said to respect dynamic feasibility.

The use of U offers recoverability from transient dynamic infeasibility: even if, at some step i, the executed state $\hat{r}(s_i)$ cannot exactly reach the planned state $r(s_i)$ and only approaches it, U can realign the system with the plan at later steps. Proposition 4.1 formalizes how recoverable the dynamics violations in the predicted states are under the low-level controller U.

Proposition 4.2 *Suppose $\gamma(\cdot)$, $r(\cdot)$ and $\hat{r}(\cdot)$ satisfy Proposition 4.1. If, for every $i \in \{1, ..., N\}$, the robot's position under $\hat{r}(s_i)$ matches exactly $\gamma(s_i)$, then the trajectory planner is said to track $\gamma(\cdot)$ while preserving dynamic feasibility.*

Deriving a finite-time BRT for a quadrotor, an underactuated non-linear system, is non-trivial. We therefore approximate the BRT via sampling. Let $\psi(x_0, x, \Delta t)$ be a procedure that simulates the closed-loop dynamics under U from initial state x_0 and returns whether the target state x is reached within time Δt. Formally, $\xi_c(x, \Delta t)$ comprises all x_0 for which $\psi(x_0, x, \Delta t)$ holds. Hence:

$$\mathbb{E}_{r(s_i), \hat{r}(s_i) \sim \gamma}\left[\psi\left(\hat{r}(s_i), r(s_{i+1}), t_i\right)\right] = Pr\Big(\hat{r}(s_i) \in \xi_c(r(s_{i+1}), t_i)\Big), \tag{5}$$

which serves as an empricial approximation to the finite-time BRT, and yields a trackability-based measure of the dynamic infeasibility of model predictions.

4.2.1 Robustness: Sensitivity of Trackability to Input Perturbations

Moreover, we define *robustness* as the sensitivity of trackability to bounded perturbations of the model inputs. As a local generalization metric, robustness encodes a Lipschitz-type property: small input variations should induce only small changes in the planned path parameterization.

Let ϵ denote the perturbation scale. Define $\pi_\epsilon(\gamma)$ as the family of geometric paths that deviate from $\gamma(\cdot)$ by at most ϵ at each discrete step while staying within the class of piece-wise polynomial paths.

Proposition 4.3 *Suppose $\gamma(\cdot)$ is a geometric path. If, for each $\hat{\gamma} \in \pi_\epsilon(\gamma)$ and $i \in \{1, ..., N-1\}$,*

$$\hat{r}(s_i) \in \xi_c(r(s_{i+1}), t_i), \text{ where } t_i = 2\Delta s/(\sqrt{h(s_i)} + \sqrt{h(s_{i+1})}), \tag{6}$$

with $r(\cdot)$, $\hat{r}(\cdot)$ and $h(\cdot)$ all corresponding to $\hat{\gamma}$, then the trajectory planner is ϵ-robust to respect dynamic feasibility.

Correspondingly, we quantify ϵ-robustness by sampling a set of perturbed geometric paths and simulating them under U based on the predicted path parameterization.

	TOPPQuad	LSTM		Transformer		ETransformer		MLP	
		Train	Test	Train	Test	Train	Test	Train	Test
max dev (m)	0.053	**0.074**	0.143	0.607	0.649	0.195	0.226	0.252	0.305
thrust violation (N)	0.000	**0.002**	0.009	0.135	0.123	0.012	0.018	0.031	0.048
TD ratio (%)	5.929(s)	-0.70%	**-0.40%**	-8.50%	-2.35%	1.89%	2.49%	-6.59%	-3.03%
failure (%)	0.0%	2.0%	4.0%	76.0%	72.0%	6.0%	4.0%	**0.0%**	6.0%
compute time (s)	10.656	0.078	0.096	1.012	1.042	0.010	0.018	**0.005**	0.014

Table 1: **Ablation study on model architectures.** Each statistic is averaged over 100 trajectories. A negative *TD ratio* ($\frac{\text{PRED. TIME}-\text{OPT. TIME}}{\text{OPT. TIME}}$) denotes shorter travel time relative to TOPPQuad (OPT. TIME). Although the low-level controller enforces per-motor limits, the nonlinear mapping from the inputs of the TOPP problem to predicted motor thrusts is not bounded by construction. Hence, the approximate nature of learning-based methods can lead to faster path execution times, albeit with non-zero thrust violations. A key desideratum of such methods is to minimize thrust violations without producing overly conservative (slow) trajectories.

4.2.2 Robustness Enhancement via Noise Injection

Proposition 4.3 naturally inspires a new training scheme that augments the dataset with randomized path perturbations. Rather than training exclusively on the original paths $\gamma(\cdot)$, we also include the perturbed paths $\hat{\gamma} \in \pi_\epsilon(\gamma)$ under a given perturbation scale ϵ, targeting to predict the same ground-truth $[h(\cdot), \cos\theta_z(\cdot)]$ at γ. To ensure practical feasibility, we adopt the following assumption:

Assumption 4.4 *Let $\gamma(\cdot)$ be a geometric path, and ϵ be a perturbation scale. For each $\hat{\gamma} \in \pi_\epsilon(\gamma)$, the control sequence $u(\cdot)$ that is optimal for γ remains ϵ-robust for $\hat{\gamma}$.*

In essence, this assumption constrains how large ϵ can be. Beyond model robustness, an excessively large ϵ would significantly compromise the time-optimal quality of the predictions. While we do not derive a precise bound on ϵ in this work, we treat it as a tunable hyperparameter, and show the proposed training scheme's effectiveness in enhancing model robustness in Section 5.1.2.

5 Experimental Results

5.1 Simulation Experiments

We construct the training dataset by generating minimum-snap trajectories through randomly sampled waypoints within a given range, discretized by 100 equally spaced points. For each path, we apply TOPPQuad to obtain a dynamically feasible, time-optimal path parameterization under a $5m/s$ speed limit, resulting in a dataset of $10,000$ trajectories. All simulation experiments are conducted in RotorPy [21] configured with CrazyFlie 2.0 parameters [22].

5.1.1 Architecture Ablation

We begin by describing the candidate architectures. **LSTM Encoder-Decoder** [23, 24] uses an LSTM encoder (with a non-parameterized attention mechanism equipped) mapping the input trajectory to a latent representation, and an LSTM decoder to generate outputs auto-regressively. **Transformer Encoder-Decoder** (denoted as *Transformer*) [25] uses self-attention in the encoder to capture intra-sequence dependencies, and cross-attention in the decoder, trained via teacher forcing [26], i.e., masking the shifted ground-truth output as the decoder input. **Encoder-Only Transformer** (denoted as *ETransformer*) removes the decoder altogether to obviate the need for teacher forcing. Finally, **Per-Step MLP** is a multilayer perceptron that predicts the outputs at each discrete step individually. Detailed implementation details and additional ablation studies appear in the Appendix.

To evaluate the trajectory planners, we measure the *maximum deviation* in position from the reference trajectories, average *thrust violation* indicating adherence to actuation constraints, and time-optimality by comparing travel Time Difference ratio (*TD ratio*) with respect to TOPPQuad. An

	LSTM			LSTM-0.01			LSTM-0.1		
ϵ (perturbation scale)	0.001	0.01	0.1	0.001	0.01	0.1	0.001	0.01	0.1
max deviation (m)	0.234	0.790	0.739	0.094	**0.093**	0.448	0.127	0.126	0.127
TD ratio (%)	-0.13%	1.33%	3.21%	0.08%	0.09%	3.73%	-0.07%	**0.04%**	0.29%
output variation	0.005	0.206	0.306	**0.000**	0.005	0.089	0.001	0.002	0.013
in-BRT probability (%)	90.0%	78.8%	70.0%	94.3%	**94.5%**	91.4%	92.4%	93.0%	92.9%

Table 2: **Robustness analysis for the LSTM Encoder-Decoder model.** Each statistic is averaged over 100 trajectories with 10 sets of perturbations randomly drawn per trajectory. *TD ratios* are computed relative to TOPPQuad's optimal solutions (not shown in the table). Non-percentage values are rounded to three decimal places, so '0.000' does not imply an exact zero.

attempt is classified as a *failure* if it leads to a crash, or if the maximum position deviation exceeds 1 meter. We also report the *compute time* required by each planner.

Table 1 presents the ablation results. **LSTM** achieves the best performance among all candidates, with a maximum position deviation only 0.023m above TOPPQuad and negligible thrust violation, resulting in an almost zero failure rate. The slight reduction in travel time arises because the LSTM occasionally yields velocities marginally above the $5m/s$ speed limit. In contrast, **Transformer** has worse tracking accuracy and higher failure rates, consistent with known difficulties in training transformers via teacher forcing with limited data [26]. This aligns with the larger TDRatio, where a faster travel time necessitates greater thrust bound violations. **ETransformer** provides competitive results but still lags behind the LSTM in tracking accuracy and time optimality. Finally, **Per-Step MLP** struggles to track the reference trajectory precisely and incurs high thrust violations as it must repeatedly base its predictions on its own prior outputs, leading to out-of-distribution issues.

5.1.2 Robustness Analysis

Next, we conduct a robustness analysis on the LSTM encoder-decoder model. We evaluate both the model trained solely on clean data (**LSTM**) and two variants, **LSTM-0.01** and **LSTM-0.1**, that incorporate randomized path perturbations of scales 0.01 and 0.1 respectively, to augment training data. To evaluate robustness, we apply controlled perturbations to the input geometric paths. Two additional metrics are reported. First, *output variation* quantifies the model sensitivity, computing the average of the maximum absolute differences in model outputs. Second, *in-BRT probability* assesses dynamic feasibility of the predicted path parameterization, as defined in Section 4.2

Table 2 presents the robustness analysis results. When trained exclusively on clean data, the model is highly sensitive to input perturbations. This is reflected by the large values in output variation. Even under small input perturbations such as $\epsilon = 0.001$, its maximal position deviation rises from 0.143 m (Table 1) to 0.234 m. Larger perturbations further degrade tracking performance and reduce in-BRT probability, which indicates a greater likelihood of generating dynamically infeasible predictions.

In contrast, the models trained with augmented noisy data yields improved output stability and in-BRT probability. Notably, **LSTM-0.1** consistently maintains low maximum position deviations and high in-BRT probabilities at all tested perturbation levels, highlighting its strong robustness. **LSTM-0.01** also shows enhanced robustness up to its training perturbation level ($\epsilon = 0.01$). Note that a higher in-BRT probability does not imply superior tracking, as it primarily measures the dynamic feasibility of the predicted trajectory under quadrotor dynamics. Furthermore, **LSTM-0.1** trades off some time-optimality and tracking accuracy for robustness, evident when examining low-perturbation regimes ($\epsilon = 0.001$). Training over a broader neighborhood of nominal paths ensures robust feasibility but can shift the solution away from the exact time-optimal trajectories. This suggests that an upper limit to training perturbation levels must exist in practice.

	TOPPQuad	LSTM	LSTM-0.1	AllocNet	MFBOTrajectory
max deviation (m)	0.039	0.124	0.170	0.037	**0.010**
min/max thrusts (N)	0.05 / 0.14	0.03 / 0.16	0.03 / 0.17	0.06 / 0.10	0.07 / 0.08
thrust violation (N)	0.000	0.002	0.002	**0.000**	**0.000**
traj time (s)	5.687	5.653	5.691	**4.066**	9.431
path length (m)	19.772	19.772	19.772	**11.453**	16.778
average speed (m/s)	3.477	**3.498**	3.474	2.817	1.779
failure rate (%)	0%	**0%**	**0%**	28%	**0%**
compute time (s)	10.083	0.064	**0.062**	0.277	13609.913

Table 3: **Baseline comparison.** Each statistic is averaged over 50 trajectories. Compute time is measured from when the target path (or waypoints) are provided until a valid path parameterization is returned. For **MFBOTrajectory**, which undergoes online retraining for each unseen waypoint configuration, the retraining time is included for a fair comparison.

5.1.3 Baseline Comparison

We compare the performance of the proposed algorithm against two learning-based baselines. Time Allocation Network (**AllocNet**) [15] processes a point cloud of obstacles to predict time-optimal trajectories for safe navigation between specified start and goal locations. Obstacles are constructed from a sequence of bounding boxes placed around the reference trajectory. Multi-Fidelity Black-Box Optimization Trajectory Planner (**MFBOTrajectory**) [11] employs Gaussian Processes to allocate time between waypoints, requiring iterative online simulations to handle unseen waypoint arrangements. These methods aim to generate time-optimal trajectories within the class of polynomials, so the resulting paths may deviate significantly from the **TOPPQuad** and **LSTM** path. Examples are provided in the Appendix. Hence, the quadrotor's *average speed* and mean *path length* are reported.

Table 3 summarizes this comparison. **AllocNet** suffers from a higher failure rate due to a mismatch between predicted time allocations and the polytopes used in the safe flight corridor during prediction. Notably, our testing setup uses generous bounding boxes to avoid distribution mismatch issues. Among its successful runs, **AllocNet**'s predicted trajectories achieve shorter travel times by finding shorter geometric paths, yet yields a lower average speed than **LSTM**. **MFBOTrajectory** achieves zero failures, but produces the slowest average speed, leading to longer travel time despite formulating shorter polynomial paths than **LSTM**. Moreover, **MFBOTrajectory** must retrain online with each new unseen waypoint arrangement, incurring significant computational overhead. Unlike the **LSTM**, neither approach utilizes the full flight profile of the quadrotor, as seen by the commanded minimum and maximum thrusts.

5.2 Hardware Experiments

Next, we validate our approach on hardware using a CrazyFlie 2.0 quadrotor tracked in a Vicon motion capture space. For safety, we limit the maximum speed to $2\ m/s$, the maximum acceleration to $10\ m/s^2$, and the maximum angular velocity to $10\ rad/s$. Under these parameters, we generate a new dataset of $9,000$ trajectories in simulation, following the same procedures used in our earlier experiments. We train an LSTM encoder-decoder model with a perturbation scale of 0.001. Experiments with other training configurations are provided in Appendix.

Table 4 shows quantitative statistics, while Figure 2 provides visualizations. We conduct tests on eight distinct geometric paths, with four trials of TOPPQuad and five trials of our proposed method. As in simulation, our method yields position deviations comparable to TOPPQuad. However, the travel time difference ratio increases. This discrepancy arises due to the difficulty of settling the quadrotor at its goal, an issue exacerbated in the learning-based approach by the sim-to-real gap.

Moreover, the proposed method naturally generalizes to longer sequences than those encountered in training. The key modification is to have the model predict only a segment of the overall trajectory at once, while conditioning on additional information capturing the robot's state at the start of each

	TOPPQuad	Learning-based TOPPQuad
max deviation (m)	0.347	0.355
travel time (s)	7.981	8.355

Table 4: Hardware experiments.

segment. As sequence lengths grow, the compute speedups become increasingly substantial. Figure 3 demonstrates the resulting extended trajectory predictions.

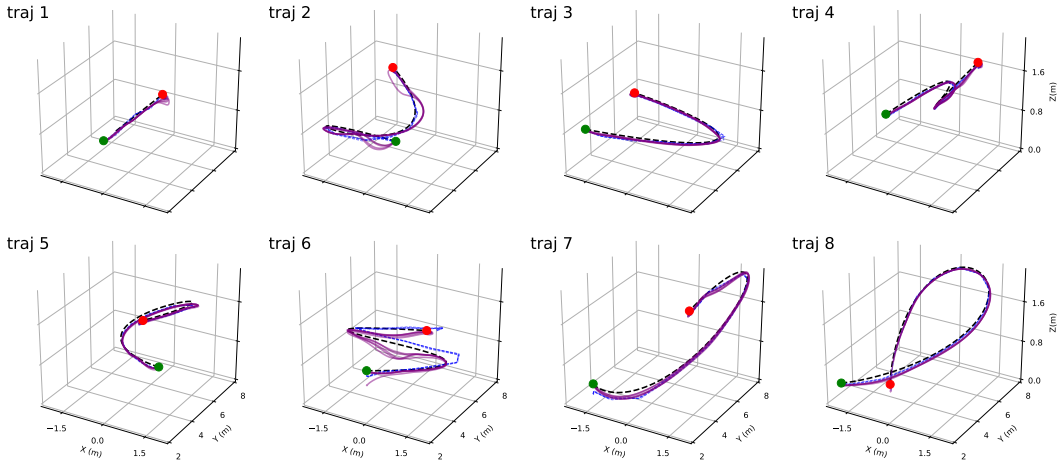
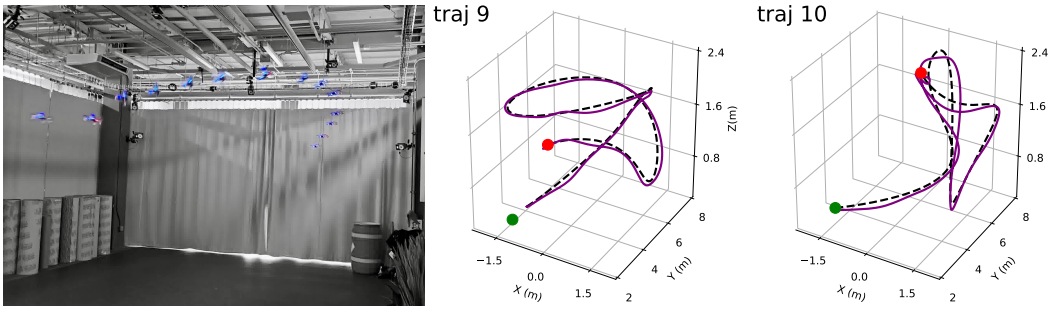


Figure 2: Experiment visualization, plotted from the collected motion capture flight data. The dashed black line represents the reference geometric path, the dashed blue line shows the tracked trajectory generated by TOPPQuad, and the solid purple lines show the tracked trajectories output by our model across different runs.



(a) (b)

Figure 3: a) Partial depiction of Traj 9. b) Experiment Visualization, showcasing good tracking of geometric paths whose lengths exceede what is seen in the dataset

6 Conclusion

In this work, we propose an imitation-learning framework for time-optimal quadrotor trajectory generation. Guided by domain-specific insights, we introduce a concise yet effective input-output feature design. We also present a rigorous robustness analysis framework alongside a data augmentation strategy that enhances model robustness. Through comprehensive studies, our method is shown to closely imitate a model-based trajectory planner, producing near-optimal solutions that largely respect dynamic feasibility and delivering a significant computational speedup over similar optimization-based methods. Finally, we validate the approach on a hardware quadrotor platform, demonstrating its practical effectiveness.

6.1 Limitations

A key limitation of the proposed approach is that the model must be retrained whenever the quadrotor's configuration changes, as its learned parameters are tailored to specific hardware setups and dynamic constraints from the training dataset. Furthermore, while the model generally approximates the expert solution accurately, the approximate nature of learning-based methods may still yield infeasible path parameterizations. Finally, the optimal perturbation bounds for data augmentation remain an open question.

References

[1] K. Mao, I. Spasojevic, M. A. Hsieh, and V. Kumar. Toppquad: Dynamically-feasible time-optimal path parametrization for quadrotors. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13136–13143, 2024. doi:10.1109/IROS58592.2024.10801611.

[2] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011. doi:10.1109/ICRA.2011.5980409.

[3] Z. Wang, X. Zhou, C. Xu, and F. Gao. Geometrically constrained trajectory optimization for multicopters. *IEEE Transactions on Robotics*, 38(5):3259–3278, 2022. doi:10.1109/TRO.2022.3160022.

[4] W. A. De Vries, M. Li, Q. Song, and Z. Sun. An alternating peak-optimization method for optimal trajectory generation of quadrotor drones. In *2024 European Control Conference (ECC)*, pages 3267–3272, 2024. doi:10.23919/ECC64448.2024.10591215.

[5] S. Liu, M. Watterson, S. Tang, and V. Kumar. High speed navigation for quadrotors with limited onboard sensing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1484–1491, 2016. doi:10.1109/ICRA.2016.7487284.

[6] S. Liu, K. Mohta, N. Atanasov, and V. Kumar. Search-based motion planning for aggressive flight in se(3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446, 2018. doi:10.1109/LRA.2018.2795654.

[7] P. Foehn, A. Romero, and D. Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021. doi:10.1126/scirobotics.abh1221. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abh1221`.

[8] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022. doi:10.1109/TRO.2022.3173711.

[9] K. Teissing, M. Novosad, R. Penicka, and M. Saska. Real-time planning of minimum-time trajectories for agile uav flight. *IEEE Robotics and Automation Letters*, 9(11):10351–10358, 2024. doi:10.1109/LRA.2024.3471388.

[10] F. Meyer, K. Glock, and D. Sayah. Top-uav: Open-source time-optimal trajectory planner for point-masses under acceleration and velocity constraints. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2838–2845, 2023. doi:10.1109/IROS55552.2023.10342270.

[11] G. Ryou, E. Tal, and S. Karaman. Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers. *The International Journal of Robotics Research*, 40(12-14):1352–1369, 2021.

[12] G. Ryou, E. Tal, and S. Karaman. Real-time generation of time-optimal quadrotor trajectories with semi-supervised seq2seq learning. In *Conference on Robot Learning*, pages 1860–1870. PMLR, 2023.

[13] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao. Learning-based motion planning in dynamic environments using gnns and temporal encoding. *Advances in Neural Information Processing Systems*, 35:30003–30015, 2022.

[14] S. Tankasala and M. Pryor. Accelerating trajectory generation for quadrotors using transformers. In N. Matni, M. Morari, and G. J. Pappas, editors, *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pages 600–611. PMLR, 15–16 Jun 2023. URL https://proceedings.mlr.press/v211/tankasala23a.html.

[15] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar. Deep learning for optimization of trajectories for quadrotors. *IEEE Robotics and Automation Letters*, 9(3):2479–2486, 2024.

[16] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212, 2021. doi:10.1109/IROS51168.2021.9636053.

[17] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[18] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, 2010. doi:10.1109/CDC.2010.5717652.

[19] M. Watterson and V. Kumar. Control of quadrotors using the hopf fibration on so (3). In *Robotics Research: The 18th International Symposium ISRR*, pages 199–215. Springer, 2019.

[20] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[21] S. Folk, J. Paulos, and V. Kumar. RotorPy: A python-based multirotor simulator with aerodynamics for education and research. *arXiv preprint arXiv:2306.04485*, 2023.

[22] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42, 2017. doi:10.1109/MMAR.2017.8046794.

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[26] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.