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], and Vijay Kumar[1]

[1]University of Pennsylvania, [2]University of California San Diego

***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 modeling architectures 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. Compared to classical planners, our method achieves substantial speedups, and we validate its real-time feasibility on a hardware quadrotor platform.**

I. Introduction

Optimal trajectory generation is one of the core components 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 key algorithmic challenge underlying synthesizing time-optimal trajectories lies in the non-convexity. A major part of non-convexity is the nonlinear nature of robot dynamics. The majority of previous approaches have used either simplified dynamics models or proxy actuation constraints. Yet another set of approaches planned trajectories with both faithful dynamics models and suitable actuation constraints. Such planners have exhibited superior mission execution time, at the cost of much higher 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 model and actuation constraints. A rigorous robustness analysis framework is proposed to quantify how well predicted trajectories is dynamically feasible, which inspires a data augmentation strategy that enhances model robustness against unseen path geometries. We perform both simulation and hardware experiments to evaluate the proposed method.

II. Related Work

The nature of time-optimal quadrotor trajectories requires plans that push the system to its physical limits. Some approaches plan trajectories following a polynomial structure, where trajectories are characterized by a set of waypoints and their time allocations [1, 2]. Other approaches plan around the full dynamics to utilize the full flight profile of the quadrotor [3, 4], but require heavy computation cost due to non-convexities, motivating the potential of learning-based solutions. [5] trains an LSTM to learn both the intermediary waypoints and time allocation for a polynomial trajectory. [6] utilizes Reinforcement Learning to generate near-time optimal trajectories for the selected drone racing tracks, but cannot guarantee saftey in cluttered environments.

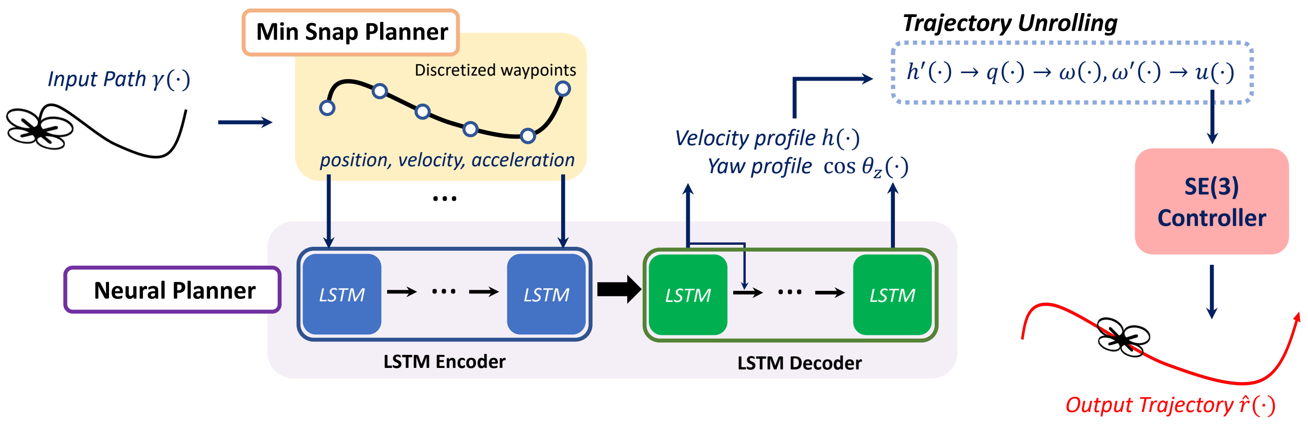
III. Preliminary

TOPPQuad [3] is an optimization-based approach for Time-Optimal Path Parameterization (TOPP) which generates dynamically feasible quadrotor trajectories to track a given 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 dynamic 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. Conversely, [7] shows that a quadrotor is a differentially flat system, 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.

IV. Methodolgy

A. Imitation Learning Problem Formulation

Given $\gamma(\cdot)$, TOPPQuad produces a time-optimal, dynamic 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 them jointly in a direct manner. We propose to use $[h(\cdot), \cos\theta_z(\cdot)]$ as the output variables of the model, where $\cos\theta_z(\cdot)$ encodes the yaw rotation encoded via the cosine function.



Min Snap Planner
Input Path $\gamma(\cdot)$
Discretized waypoints
position, velocity, acceleration
Neural Planner
LSTM
LSTM Encoder
LSTM Decoder
Velocity profile $h(\cdot)$
Yaw profile $\cos\theta_z(\cdot)$
Trajectory Unrolling
$h'(\cdot) \to q(\cdot) \to \omega(\cdot), \omega'(\cdot) \to u(\cdot)$
SE(3) Controller
Output Trajectory $\hat{r}(\cdot)$

Fig. 1. Overall pipeline. We begin by discretizing the input geometric path $\gamma(\cdot)$ into equally spaced grid points using a minimum-snap planner. The trained neural planner then predicts squared-speed and yaw profiles, imitating the model-based time-optimal planner TOPPQuad [3]. The full robot state trajectory is unrolled and used by the low-level geometric controller to compute control inputs.

Next, we discuss how to recover the original variables. 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, aligning the drone's body-z axis with b_3 and setting yaw to the desired θ_z. Next, we calculate rotation, yielding angular velocity ω and its derivative ω'. Finally, a low-level geometric controller [8] computes $u(\cdot)$ from the derived states.

We augment the model's input with path curvature, i.e., the first and second derivatives of the geometric path $\gamma'(\cdot)$ and $\gamma''(\cdot)$, to provide explicit geometric information. In summary, we formulate the imitation task as learning the mapping:

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

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

B. Robustness Analysis

To ensure effective generalization, it is crucial to thoroughly evaluate the robustness of the learned models beyond 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 $r(\cdot) := \{h(\cdot), q(\cdot), \omega(\cdot)\}$. Let $\hat{r}(\cdot)$ denote the full robot state trajectory after executing the error-tracking low-level controller U [8]:

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

which integrates the quadrotor dynamics $\dot{f}$ over the spatial step-size Δs. To formalize robustness, we introduce the concept of Backward Reachable Tube (BRT) [9]: Let $\xi_U(x, \Delta t)$ denote the subset of the state space from which state x can be reached within Δt seconds under U:

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

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

$$\hat{r}(s_i) \in \xi_U(r(s_{i+1}), t_i), \tag{4}$$

where $t_i = 2\Delta s/(\sqrt{h(s_i)} + \sqrt{h(s_{i+1})})$, then the trajectory planner is robust with respect to dynamic feasibility.

Proposition IV.1 captures how violations in the planned state are recoverable under the low-level controller. Even more, if $\hat{r}(\cdot)$ coincides with $\gamma(\cdot)$, then the tracking problem is solved.

Proposition IV.2. *Suppose $\gamma(\cdot)$, $r(\cdot)$ and $\hat{r}(\cdot)$ satisfy Proposition IV.1. If, for each $i \in \{1, ..., N\}$, the robot's coordinate under $\hat{r}(s_i)$ matches exactly $\gamma(s_i)$, then the trajectory planner is robust for tracking $\gamma(\cdot)$ while respecting dynamic feasibility.*

Deriving a finite-time BRT for the quadrotor, an under-actuated non-linear system, is non-trivial. In this work, we approximate the BRT via a sampling-based approach. Let $\psi(x_0, x, \Delta t)$ be a procedure that applies the low-level controller U from initial state x_0 to verify whether it can reach target state x within time Δt. Formally, $\xi_c(x, \Delta t)$ comprises all x_0 for which $\psi(x_0, x, \Delta t)$ holds. Hence:

$$\begin{aligned}&\mathbb{E}_{r(s_i), \hat{r}(s_i) \sim \gamma}\Big[\psi\Big(\hat{r}(s_i), r(s_{i+1}), t_i\Big)\Big] \\ &\qquad = Pr\Big(\hat{r}(s_i) \in \xi_c(r(s_{i+1}), t_i)\Big).\end{aligned} \tag{5}$$

We estimate this probability by sampling $r(s_i)$ and $\hat{r}(s_i)$ from the model predictions and simulating U to determine whether $\hat{r}(s_i)$ can reach $r(s_{i+1})$ within t_i. However, the sequence space of γ is prohibitively large, making it hard to obtain an unbiased probability estimate.

A more practical measure of planner robustness is the variation in tracking robustness under small perturbations to

	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 I
ABLATION STUDY ON MODEL ARCHITECTURES (100 TRIALS).

the input paths. 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 IV.3. *Suppose $\gamma(\cdot)$ is a geometric path. If, for each $\hat{\gamma} \in \pi_\epsilon(\gamma)$ and $i \in \{1, ..., N-1\}$,*

$$\hat{r}(s_i) \in \xi_c(r(s_{i+1}), t_i), \tag{6}$$

where $t_i = 2\Delta s / (\sqrt{h(s_i)} + \sqrt{h(s_{i+1})})$ and $r(\cdot)$, $\hat{r}(\cdot)$ and $h(\cdot)$ all correspond to $\hat{\gamma}$, then the trajectory planner is ϵ-robust with respect to dynamic feasibility.

C. Robustness Enhancement via Noise Injection

Proposition IV.3 inspires a new training scheme that augments the dataset with randomized path perturbations. Instead of 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 IV.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 $\hat{\gamma}$.*

V. EXPERIMENTAL RESULTS

A. Simulation Experiments

We generate the training dataset from $10,000$ TOPPQuad trajectories bounded to $5m/s$ with minimum snap paths through randomized waypoints. All simulation experiments are conducted in RotorPy with CrazyFlie 2.0 parameters.

1) Architecture Ablation: We begin by describing the candidate architectures. **LSTM Encoder-Decoder** 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*) uses self-attention in the encoder to capture intra-sequence dependencies, and cross-attention in the decoder, trained via teacher forcing. **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.

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 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 I 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. 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.

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 and two variants, **LSTM-0.01** and **LSTM-0.1**, that incorporate randomized path perturbations of scales 0.01 and 0.1, respectively, as 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 IV-B

Table II presents the robustness analysis results. When trained exclusively on clean data, the model is highly sensitive to input perturbations. Under $\epsilon = 0.001$, its maximal position deviation rises. 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

	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 II
ROBUSTNESS ANALYSIS FOR THE LSTM ENCODER-DECODER MODEL (10 PERTURBATIONS PER TRIAL ACROSS 100 TRIALS).

	TOPPQuad	LSTM	LSTM-0.01
max deviation (m)	0.347	0.355	0.372
travel time (s)	7.981	8.355	8.114

TABLE III
HARDWARE EXPERIMENTS (40 TRIALS).

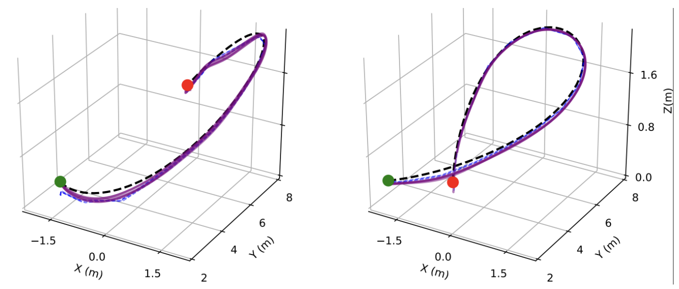


Fig. 2. Experiment visualization, plotted from the collected flight data. The dashed black line represents the reference geometric path. The dashed blue and solid purple lines show the tracked trajectories generated by TOPPQuad and by our method, respectively.

data yield improved output stability and in-BRT probability. **LSTM-0.1** consistently maintains low maximum position deviations and high in-BRT probabilities. **LSTM-0.01** also shows enhanced robustness up to its training perturbation level. Note that **LSTM-0.1** trades off some tracking accuracy for robustness, evident when examining low-perturbation regimes.

B. 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. Table III shows quantitative statistics, while Figure 2 provides visualizations. Our method yields position deviations comparable to TOPPQuad, while the travel time is suboptimal. Notably, incorporating randomized path perturbations into the proposed model leads to improved travel time.

VI. CONCLUSION

In this work, we propose an imitation-learning framework for time-optimal quadrotor trajectory generation. 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.

REFERENCES

[1] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar. High speed navigation for quadrotors with limited onboard sensing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1484–1491, 2016.

[2] Sikang Liu, Kartik Mohta, Nikolay Atanasov, and Vijay Kumar. Search-based motion planning for aggressive flight in se(3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446, 2018.

[3] Katherine Mao, Igor Spasojevic, M. Ani Hsieh, and Vijay 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.

[4] Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022.

[5] Yuwei Wu, Xiatao Sun, Igor Spasojevic, and Vijay Kumar. Deep learning for optimization of trajectories for quadrotors. *IEEE Robotics and Automation Letters*, 9(3):2479–2486, 2024.

[6] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[7] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

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

[9] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire 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.