Real-Time Generation of Time-Optimal Quadrotor Trajectories with Semi-Supervised Seq2Seq Learning

Gilhyun Ryou, Ezra Tal and Sertac Karaman
Laboratory for Information and Decision Systems (LIDS),
Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139
{ghryou, eatal, sertac}@mit.edu

Supplementary Material

A video of the experiments is available at https://youtu.be/lCD6UH8G7AY.

Appendix A Preliminaries

This appendix contains additional background and related works on quadrotor trajectory planning.

In order to generate a fast and agile trajectory for a real quadrotor vehicle, an optimization problem with complex feasibility constraints arising from aerodynamic and electromechanical phenomena must be solved. By reformulating the optimization problem so that feasibility is the objective rather than a constraint, the minimum-snap method sidesteps the troublesome feasibility constraints [1, 2]. By minimizing the fourth temporal derivative of the position, i.e., the snap, this method generates smooth piece-wise polynomial trajectories that are less likely to activate the feasibility constraints.

Numerous algorithms aim to improve upon the polynomial minimum-snap trajectories, e.g., generating better optimized solutions or decreasing the required computation time. For instance, an improved solution can be found by using a more general trajectory representation, without the topological constraints of a piece-wise polynomial representation, as shown by Foehn et al. [3]. The computation time of the minimum-snap method can be reduced by avoiding non-linear optimization subject to flight dynamics constraints. For example, Gao et al. [4] separate spatial and temporal optimization to obtain a convex optimization problem subject to simple linear velocity and acceleration constraints. Similarly, Romero et al. [5] present a sampling-based method that generates a velocity search graph between two waypoints (or gates) and finds the optimal velocity profile with Dijkstra search. Alternatively, trajectory generation may be divided into global and local planning problems, where accurate vehicle dynamics are only considered locally, e.g., using model predictive control [6] or deep neural networks [7]. Visual input data can also be streamed to the local trajectory planner to avoid obstacles, even when the vehicle is in the field [8, 9]. However, a reference path that accurately guides the local planner towards the optimal global trajectory, such as the one provided by our proposed algorithm, is still required.

When compared to minimum-snap trajectory optimization, most existing works either improve the quality of the generated trajectories or reduce the computation time, but fail to do both. For instance, an algorithm may find better solutions, but impose large computational burden, even taking several hours per trajectory. On the other hand, algorithms that achieve real-time performance by adopting coarse dynamics models may result in infeasible or overly conservative trajectories. Our work considers the problem of online time-optimal trajectory planning that can re-generate a global path in real-time while maintaining a sophisticated model of the vehicle dynamics.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

Appendix B Implementation Details

In this appendix, we provide additional, more detailed description of the model and learning implementation.

B.1 Seq2Seq Model

The training dataset consists of 10,000 waypoint sequences for each sequence length in the range from five to fourteen waypoints. The dataset is augmented by flipping the x, y, and z planes, which increases the size of the dataset by a factor of eight. Since the flipped trajectories share the same minimum-snap time allocation ratio, we only solve the nonlinear optimization (4) once for each original sequence. The minimum-snap trajectory duration T_{MS} does differ between flipped trajectories and is found separately for all augmented data sequences using the method described in Section 2.1. The validation dataset consists of 200 waypoint sequences for each length and is generated with the same method.

Our proposed model is composed of encoder, decoder, VAE and attention modules. For the forward-facing trajectory, which includes a yaw reference, the encoder input consists of the waypoints, normalized to $[-1, 1]^4$, as follows:

$$x^{\text{in}}_{p,i} = [\tilde{p}^i_r/(L_{\text{space}}/2) \;\; \cos(\tilde{p}^i_\psi) \;\; \sin(\tilde{p}^i_\psi) \;\; f_{\text{EOS}}], \tag{1}$$

where $\tilde{p}^i_r$ and $\tilde{p}^i_\psi$ are the position and yaw of the i-th waypoint, respectively. The indicator function f_{EOS} is unity for the final waypoint in the sequence and zero otherwise. For the constant yaw trajectory, which does not include a yaw reference, the input is reduced to

$$x^{\text{in}}_{p,i} = [\tilde{p}^i_r/(L_{\text{space}}/2) \;\; f_{\text{EOS}}]. \tag{2}$$

For both types of input, we denote the dimension of each waypoint by d_{wp}. We use a bidirectional GRU with a hidden layer size of 256 as the encoder (i.e., $d_{\text{enc}} = 2 \times 256$), and a basic GRU with a hidden layer size of 256 as the decoder (i.e., $d_{\text{dec}} = 256$).

The VAE estimates the mean μ_{VAE} and variance σ_{VAE} of the latent vector

$$z_{\text{VAE}} = \mu_{\text{VAE}} + \epsilon \odot \sigma_{\text{VAE}} \tag{3}$$

with $\epsilon \sim \mathbb{N}(0, I)$. The encoder output, with size d_{enc}, is first converted to the decoder size d_{dec} using a 512×256 fully-connected layer and rectified linear unit (ReLU). The mean μ_{VAE} and variance σ_{VAE} are each generated from the converted hidden encoder output using separate fully-connected networks consisting of 256×32 and 32×32 hidden layers with ReLU in between.

The decoder is guided using attention information generated from the hidden encoder states. First, a score value, which represents the correlation between the hidden encoder and decoder states, is estimated as follows:

$$\tilde{\mathbf{a}}_{i,j} = \mathbf{v}_{\text{att}} \tanh(\mathbf{W}_{\text{att}}[\mathbf{h}_{\text{enc},j}; \mathbf{h}_{\text{dec},i}]), \tag{4}$$

where $\mathbf{h}_{\text{enc},j} \in \mathbb{R}^{d_{\text{enc}}}$ is the j-th hidden encoder state and $\mathbf{h}_{\text{dec},i} \in \mathbb{R}^{d_{\text{dec}}}$ is the i-th hidden decoder state. The attention module also uses a weight matrix $\mathbf{W}_{\text{att}} \in \mathbb{R}^{d_{\text{dec}} \times (d_{\text{enc}} + d_{\text{dec}})}$ and energy vector $\mathbf{v}_{\text{att}} \in \mathbb{R}^{1 \times (d_{\text{dec}})}$. The score values are normalized using the softmax function

$$\mathbf{a}_{i,j} = \exp(\tilde{\mathbf{a}}_{i,j}) / \sum_{j'} \exp(\tilde{\mathbf{a}}_{i,j'}), \tag{5}$$

and the attention is obtained as the weighted sum of the hidden encoder states

$$\mathbf{a}_i = \sum_j \mathbf{a}_{i,j} \mathbf{h}_{\text{enc},j}. \tag{6}$$

The attention information, the previous hidden state, and the previous outputs are concatenated to obtain the input for the basic GRU decoder. Hence, the input dimension of the decoder is $d_{\text{wp}} + d_{\text{enc}} + d_{\text{dec}}$. The initial decoder input is the hidden state reconstructed from the VAE and the first waypoint $x^{\text{in}}_{p,0}$.

In the first and second training phases, the proposed seq2seq model is trained by supervised learning using the Adam optimizer with a 1×10^{-3} learning rate. This learning rate is decayed by a factor of 0.995 at every epoch during the first phase, and by a factor of 0.9999 at every epoch during the second phase. In the third training step, we further optimize the model using reinforcement learning and use the same Adam optimizer with a 1×10^{-4} learning rate. During this final phase, the learning rate is decayed by a factor of 0.995 at every epoch. The RL reward decay γ is set to 0.9, and the action variance σ_{rf} is set to 0.1.

B.2 Bayesian Optimization

Before augmenting the training dataset, we select 200 waypoint sequences for each sequence length to form the BO dataset. For each sample, the time-optimal time allocation is found using BO based on (6). The basic GPC is used as a surrogate model to approximate the boundary of the feasibility set $\mathcal{P}_T$. Given a set of data points $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ with corresponding evaluations $\mathbf{y} = \{y_1, \cdots, y_N\}$, GPC assumes a joint Gaussian distribution of the evaluations and the latent variables $\mathbf{f} = [f_1, \cdots, f_N]$ and predicts the probability $P(y_*|\mathbf{y}, \mathbf{x}_*, \mathbf{X})$ for a test point $\mathbf{x}_*$ based on these latent variables. The latent variables and the hyperparameters of the kernel function are trained by maximizing the marginal likelihood function

$$P(\mathbf{y}, \mathbf{f}|\mathbf{X}) = \Pi_{i=n}^{N} P(y_n|f_n)P(\mathbf{f}|\mathbf{X}) = \Pi_{n=1}^{N} \mathcal{B}(y_n|\Phi(f_n))\mathcal{N}(\mathbf{f}|0, K(\mathbf{X}, \mathbf{X})), \tag{7}$$

where $\mathcal{B}(x)$ is the Bernoulli likelihood and $\Phi(f_n)$ is the cumulative density function used to map the latent variable f_n onto the probability domain [0, 1]. The covariance kernel $K(\mathbf{X}, \mathbf{X})$ is built based on the Gaussian prior assumption. The covariance between $\mathbf{X}$ and a test point $\mathbf{x}_*$ is modeled with the same kernel and the resulting class probability is obtained as

$$P(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int P(y_*|f_*)P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})d\mathbf{f}_*. \tag{8}$$

Each evaluation point is selected as the maximum of the *acquisition function* $\alpha(\mathbf{x}|\mathcal{D})$, which balances uncertainty reduction in the surrogate model against the anticipated improvement in the objective function, given all data $\mathcal{D}$ obtained in previous evaluations. The BO surrogate model is initialized using 1024 evaluations around an initial solution obtained from minimum-snap optimization. For this initialization dataset, we sample points until we have obtained 512 evaluations on each side of the feasibility boundary.

In the simulated environment, we improve overall optimization efficiency by using multi-fidelity BO, which combines cheap low-fidelity evaluation with expensive high-fidelity measurements. The acquisition process is also modified to account for the different fidelity levels. It provides not only the next evaluation point, but also the fidelity level l of the next evaluation, as follows:

$$\mathbf{x}_{\text{next}}, l_{\text{next}} = \underset{\mathbf{x}\in\mathcal{X}, l\in\{l^1,\ldots,l^L\}}{\arg\max} \alpha(\mathbf{x}, l|\mathcal{D}), \tag{9}$$

where $\mathcal{D} = D_{l^1} \cup \cdots D_{l^L}$ contains all past evaluations. The acquisition function itself is modified by introducing weights based on the evaluation cost at the different fidelity levels. In practice, this makes the algorithm less likely to select high-fidelity evaluations, so that the overall cost of the experiments is minimized.

The acquisition function considers exploration and exploitation. For exploration, it selects the most uncertain sample near the classifier decision boundary to maximize the effectiveness in improving the surrogate model, as follows:

$$\alpha_{\text{explore}}(\mathbf{x}, l) = -\frac{|\mu_l(\mathbf{x})|}{\sigma_l(\mathbf{x})} C_l, \tag{10}$$

where μ_l and σ_l are respectively the mean and standard deviation of the posterior distribution estimated from the surrogate model at the l-th fidelity level. The weight C_l reflects the cost of an evaluation at fidelity level l. We set $C_L = 1.0$ for the low-fidelity evaluation using the differential flatness motor speed check, and we set $C_H = 10.0$ for the high-fidelity evaluation using the 6DOF simulation. For exploitation, our implementation uses modified expected improvement with constraints (EIC) to consider both the probability of success and the corresponding variance. The resulting acquisition function is given by:

$$\alpha_{\text{exploit}}(\mathbf{x}, l) = \begin{cases} \alpha_{\text{EI}}(\mathbf{x})\tilde{P}_l(y = 1|\mathbf{x}), & \text{if } \tilde{P}_l(y = 1|\mathbf{x}) \geq h_l \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where h_l is based on the cost of an infeasible evaluation, $\tilde{P}_l(y=1|\mathbf{x}) = P_l(\mu_l(\mathbf{x}) - \beta\sigma_l(\mathbf{x}) \geq 0|\mathbf{x})$, and β is a penalty on variance. Since the objective function is deterministic, so is the expected improvement, i.e., $\alpha_{\text{EI}}(\mathbf{x}) = \sum_i \bar{x}_i - \sum_i x_i$ with $\bar{\mathbf{x}}$ the current best solution. For both low-fidelity and high-fidelity evaluation, we use $h_l = 1.0$ and $\beta = 3.0$. The final acquisition function

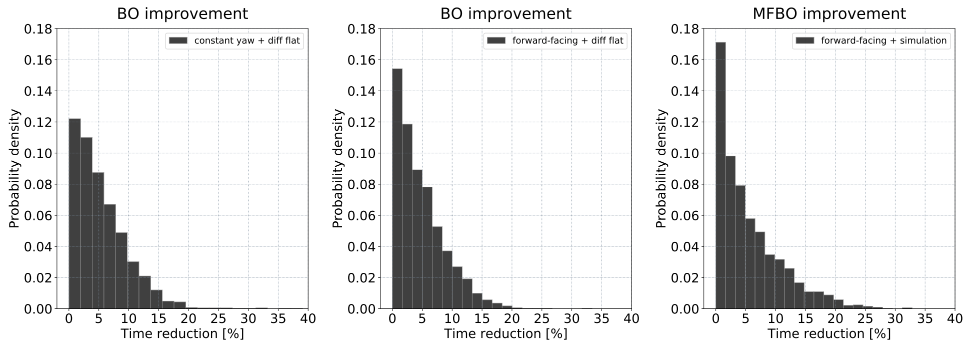
$$\alpha(\mathbf{x}, l) = \begin{cases} \alpha_{\text{exploit}}(\mathbf{x}, l), & \text{if } \exists x \in \mathcal{X} \text{ s.t. } \tilde{P}_l(y=1|\mathbf{x}) \geq h_l \\ \alpha_{\text{explore}}(\mathbf{x}, l), & \text{otherwise} \end{cases} \tag{12}$$

performs exploitation only if there is sufficient confidence in its feasibility.

For each single-fidelity BO iteration, 128 samples are acquired for evaluation based on the reference motor speeds from the differential flatness transform. For multi-fidelity BO, 64 samples are acquired in differential-flatness-based low-fidelity iterations and at most four samples are acquired in high-fidelity iterations using the 6DOF simulation. In order to curb the computational cost, we limit BO to 30 iterations for each waypoint sequence. As described in Section 4, the fully-trained seq2seq model outperforms the BO labels, showing that BO may not always be converged at termination. Despite this lack of convergence, BO successfully guides the final RL learning phase and attains significant improvements when compared to minimum-snap optimization, as shown in Table 1 and Fig. 1.

Table 1: Average trajectory time reduction for BO labels compared to minimum-snap labels.

Feasibility constraint	Differential flatness		6DOF Simulation
Yaw reference	Constant	Forward-facing	Forward-facing
Reduction	5.375 %	4.746 %	5.552 %

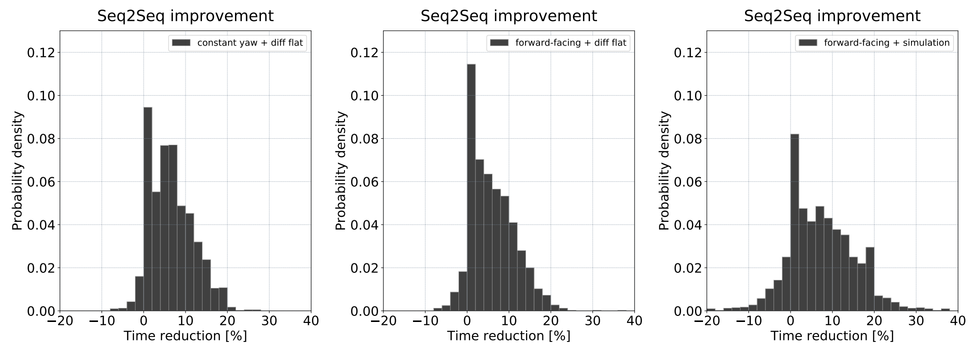


(a) Constant yaw + Diff. flat (b) Forward-facing + Diff. flat (c) Forward-facing + Simulation

Figure 1: Trajectory time reduction for BO labels compared to minimum-snap labels.

Appendix C Seq2Seq Learning Results

This appendix contains detailed experimental results. As described in Section 4, we trained two types of models: (i) using analytical feasibility evaluations based on the reference motor speeds obtained from the simplified quadrotor dynamics differential flatness transform, and (ii) using numerical evaluations of trajectory-tracking accuracy in a six-degree-of-freedom (6DOF) flight dynamics simulation. For the training with differential flatness feasibility constraints, we consider two different datasets: one with only waypoint positions and no yaw (i.e., constant yaw at zero), and one with a tangential (i.e., forward-facing) yaw reference. Hence, we evaluated our algorithm in three different training setups: constant yaw trajectory with differential flatness feasibility constraint, forward-facing yaw trajectory with differential flatness feasibility constraint, and forward-facing yaw trajectory with 6DOF simulation feasibility constraint. Figure 2 shows the trajectory time reduction achieved in each of these training setups.



(a) Constant yaw + Diff. flat (b) Forward-facing + Diff. flat (c) Forward-facing + Simulation

Figure 2: Trajectory time reduction for optimized seq2seq trajectories compared to baseline minimum-snap trajectories for the validation dataset.

In Fig. 3, we compare the average improvement for several trajectory properties, namely the number of waypoints, the trajectory length of the baseline minimum-snap trajectory, and the ratio of the smoothness cost of the optimized seq2seq trajectory and the baseline minimum-snap trajectory. Aligning with intuition, trajectories with great length and a large number of waypoints provide more opportunity for improvement. We also observe that in most cases the optimized trajectory has an increased smoothness cost (i.e., > 1 in Fig. 3), indicating a more aggressive trajectory.

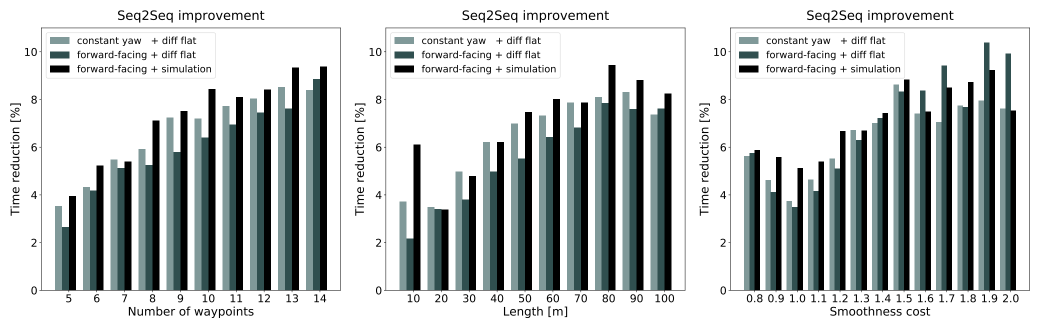
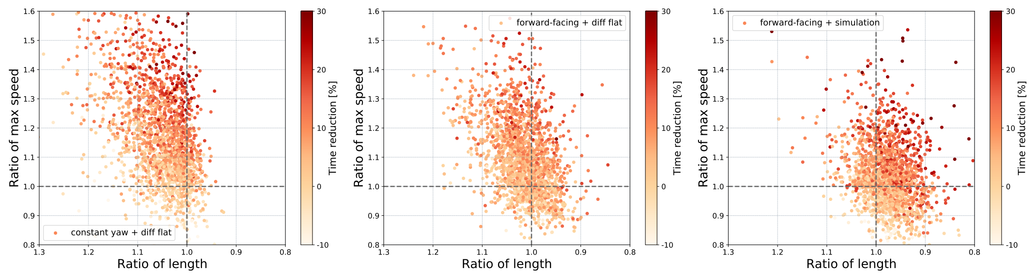


Figure 3: Trajectory time reduction for optimized seq2seq trajectories compared to baseline minimum-snap trajectories plotted against trajectory properties.

Our proposed model chiefly relies on two mechanisms to reduce the trajectory time. It generates lengthier paths that can be flown at a higher speed or lowers the flight speed to allow tighter turns. The application of these mechanisms and the corresponding trajectory time improvement are shown in Fig. 4. For the trajectories with constant yaw subject to differential flatness feasibility constraints (Fig. 4a), we observe that the trained model tends to generate longer paths that enable vehicle to fly at a higher speed. When using the 6DOF simulation to check feasibility (Fig. 4c), the model learns that the flight controller can stabilize the trajectory tracking within acceptable error bounds, even if the motor speeds are somewhat saturated. Learning the controller performance allows the model to further exploit the vehicle capabilities by planning tighter paths between waypoints, which results in more aggressive turns and lowers the trajectory time.



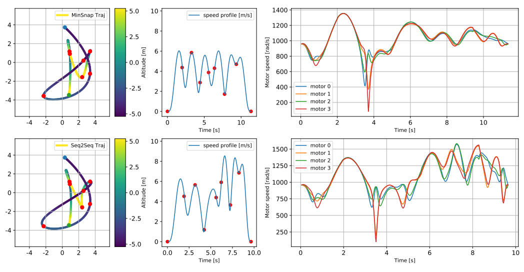
(a) Constant yaw + Diff. flat (b) Forward-facing + Diff. flat (c) Forward-facing + Simulation

Figure 4: Average time reduction over the trajectory length ratio and maximum speed ratio between the baseline and optimized trajectories.

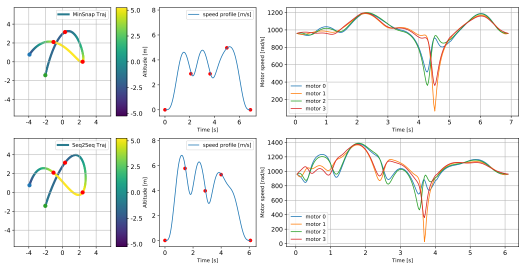
Table 2, 3, 4, and Fig. 5, 6, 7 list and visualize the trajectory time reductions along with other data from the three different training setups. We compare the duration, length, and top speed of the trajectories obtained with the baseline minimum-snap method to those obtained with our method. For the training in simulation, we also compare the seq2seq model trained with the differential flatness feasibility constraints to the seq2seq model trained with the 6DOF simulation feasibility constraints.

Table 2: Trajectory time reduction by the seq2seq (s2s) model relative to minimum-snap (MS) trajectories for the validation dataset with constant yaw and using the differential flatness feasibility constraints. The bottom eight rows contain data for the sample (i.e., the waypoint sequence and corresponding trajectories) at the percentile rank.

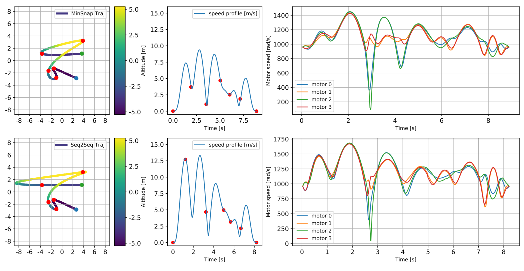
Percentile Time reduction		95th 14.99 %	75th 11.78 %	50th 7.59 %	25th 4.19 %	5th 0.73 %	1st -3.63 %
Waypoints		9	5	7	9	6	10
Duration	MS	11.4 s	6.9 s	8.9 s	12.4 s	7.0 s	12.9 s
	s2s	9.6 s	6.1 s	8.2 s	11.9 s	6.9 s	13.4 s
Top speed	MS	6.0 m/s	5.1 m/s	9.4 m/s	6.4 m/s	8.2 m/s	8.3 m/s
	s2s	8.5 m/s	6.8 m/s	13.3 m/s	7.3 m/s	7.9 m/s	8.4 m/s
Length	MS	43.5 m	21.3 m	37.6 m	50.5 m	23.9 m	55.0 m
	s2s	43.4 m	22.9 m	47.0 m	50.7 m	23.6 m	60.2 m
Smoothness cost	(s2s / MS)	2.3	1.8	1.7	1.1	1.0	1.3



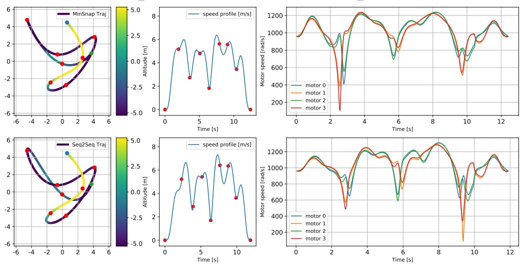
(a) 95th percentile (14.99 % improvement)



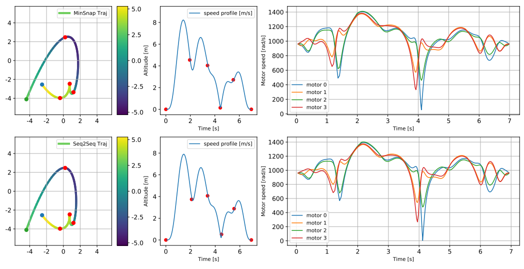
(b) 75th percentile (11.78 % improvement)



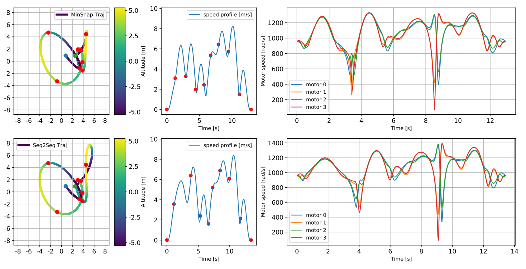
(c) 50th percentile (7.59 % improvement)



(d) 25th percentile (4.19 % improvement)



(e) 5th percentile (0.73 % improvement)

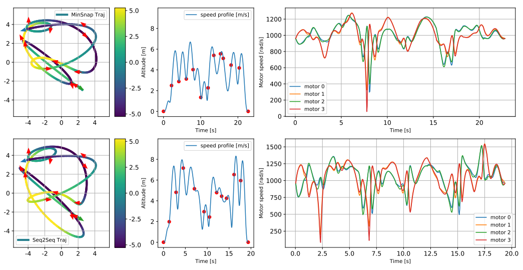


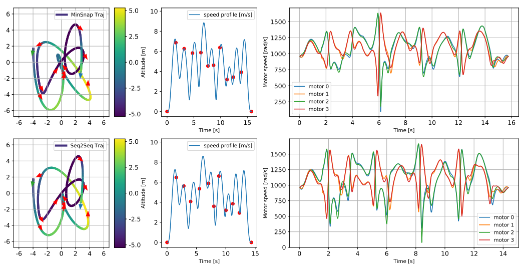
(f) 1st percentile (- 3.63 % improvement)

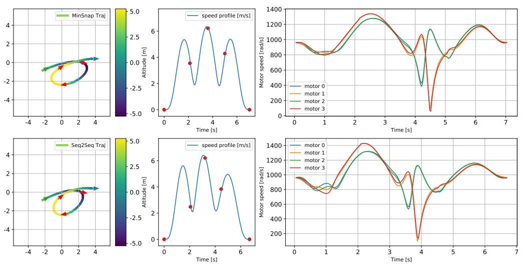
Figure 5: Comparison of the path, speed profile, and reference motor speeds of the minimum-snap baseline trajectory and the optimized seq2seq trajectory with constant yaw and using the differential flatness feasibility constraints.

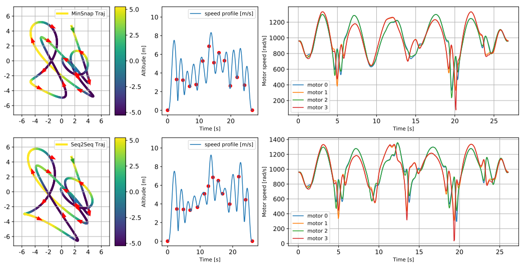
Table 3: Trajectory time reduction by the seq2seq (s2s) model relative to minimum-snap (MS) trajectories for the validation dataset with forward-facing yaw and using the differential flatness feasibility constraints. The bottom eight rows contain data for the sample (i.e., the waypoint sequence and corresponding trajectory) at the percentile rank.

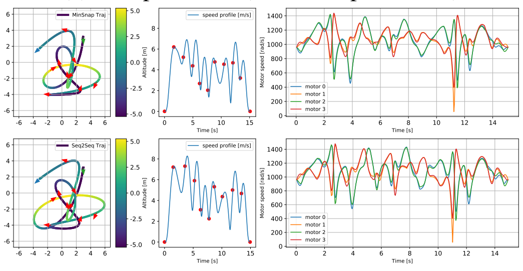
Percentile Time reduction		95th 14.94 %	75th 8.88 %	50th 4.89 %	25th 2.89 %	7th 1.08 %	1st -3.79 %
Waypoints		13	12	5	14	11	7
Duration	MS	22.8 s	15.8 s	7.1 s	26.8 s	15.9 s	9.1 s
	s2s	19.4 s	14.4 s	6.7 s	26.1 s	14.9 s	9.4 s
Top speed	MS	6.7 m/s	8.8 m/s	6.3 m/s	6.9 m/s	6.9 m/s	9.1 m/s
	s2s	8.3 m/s	8.6 m/s	6.4 m/s	8.3 m/s	7.7 m/s	8.4 m/s
Length	MS	83.4 m	71.9 m	23.2 m	115.5 m	62.1 m	38.5 m
	s2s	87.8 m	67.1 m	21.9 m	122.1 m	67.7 m	39.5 m
Smoothness cost	(s2s / MS)	1.6	0.8	1.1	1.2	1.2	1.2


(a) 95th percentile (14.94 % improvement)


(b) 75th percentile (8.88 % improvement)


(c) 50th percentile (4.89 % improvement)


(d) 25th percentile (2.89 % improvement)


(e) 7th percentile (1.08 % improvement)

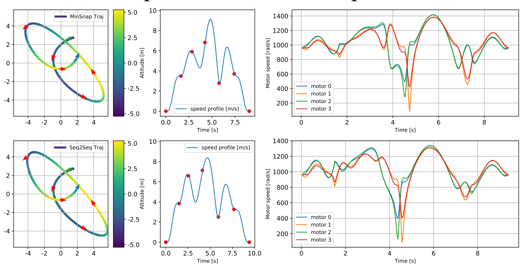
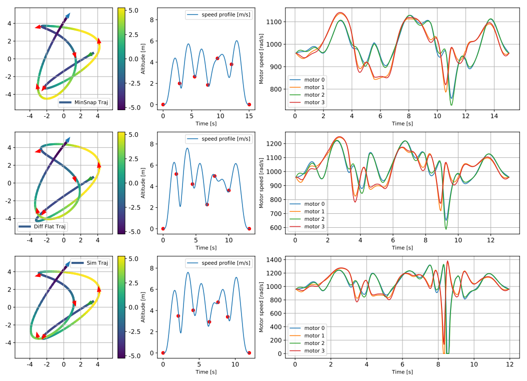
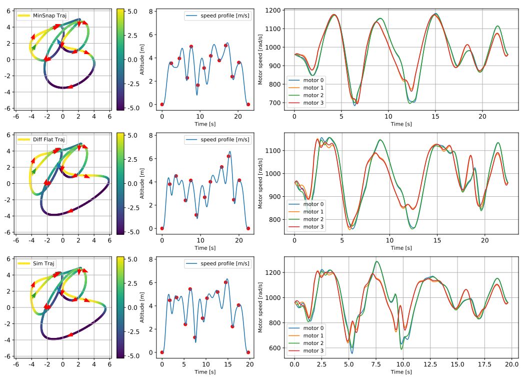

(f) 1st percentile (-3.79 % improvement)

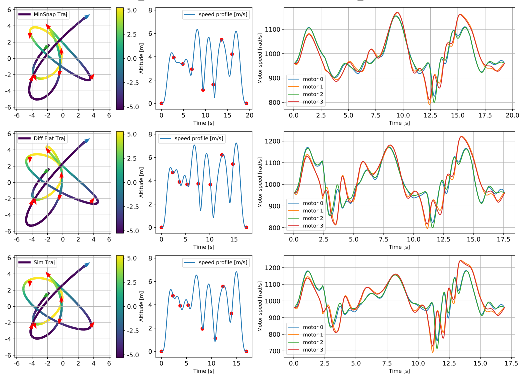
Figure 6: Comparison of the path, speed profile, and reference motor speeds of the minimum-snap baseline trajectory and the optimized seq2seq trajectory with forward-facing yaw and using the differential flatness feasibility constraints.

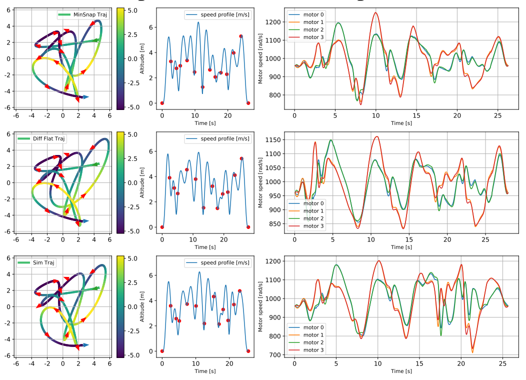
Table 4: Trajectory time reduction by the seq2seq (Sim) model relative to minimum-snap (MS) trajectories for the validation dataset with forward-facing yaw and using the 6DOF simulation feasibility constraints. The bottom four rows contain data for the sample (i.e., the waypoint sequence and corresponding trajectory) at the percentile rank. The trajectory obtained by the seq2seq model using the the differential flatness feasibility constraints (DF) is listed as well.

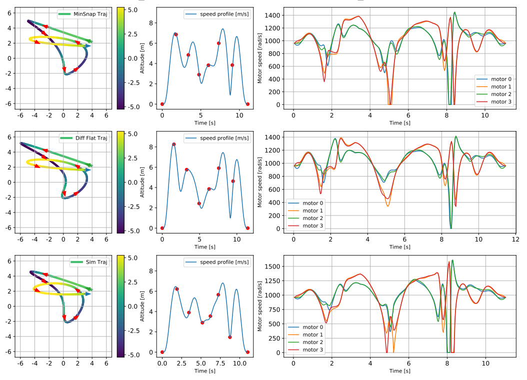
Percentile		95th	75th	50th	25th	13th	1st
Time reduction		20.52 %	13.50 %	12.06 %	2.34 %	0.36 %	-9.18 %
Waypoints		7	13	9	14	8	8
Duration	MS	15.0 s	22.7 s	19.3 s	26.3 s	11.1 s	13.4 s
	DF	13.1 s	22.4 s	17.6 s	28.0 s	11.5 s	14.1 s
	Sim	12.0 s	19.6 s	17.0 s	25.6 s	11.0 s	14.6 s
Top speed	MS	6.0 m/s	5.3 m/s	6.4 m/s	6.4 m/s	7.4 m/s	7.7 m/s
	DF	7.6 m/s	6.6 m/s	7.2 m/s	5.9 m/s	8.4 m/s	7.1 m/s
	Sim	7.6 m/s	6.3 m/s	6.8 m/s	6.3 m/s	6.8 m/s	6.5 m/s
Length	MS	51.2 m	73.7 m	67.8 m	88.2 m	46.0 m	54.7 m
	DF	54.3 m	76.9 m	72.3 m	94.6 m	49.0 m	56.4 m
	Sim	51.1 m	74.7 m	67.2 m	91.0 m	43.7 m	55.3 m
Smoothness cost	(Sim / MS)	1.1	1.4	1.2	1.1	0.9	1.1


(a) 95th percentile (20.52 % improvement)


(b) 75th percentile (13.50 % improvement)


(c) 50th percentile (12.06 % improvement)


(d) 25th percentile (2.34 % improvement)


(e) 13th percentile (0.36 % improvement)

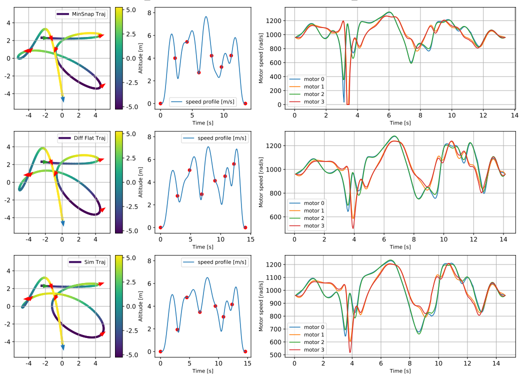

(f) 1st percentile (- 9.18 % improvement)

Figure 7: Comparison of the path, speed profile, and reference motor speeds of the minimum-snap baseline trajectory and the optimized seq2seq trajectory with forward-facing yaw and using the 6DOF simulation feasibility constraints.

Appendix D Real-World Flight Experiments

We randomly selected ten waypoint sequences to test in flight experiments using a quadrotor. Figure 8 shows the paths of the corresponding baseline minimum-snap trajectories and optimized seq2seq trajectories. During the flight experiments, we uniformly scale the corresponding time allocations until the tracking error constraints are active. Table 5 contains the trajectory time reduction of the resulting seq2seq trajectories.

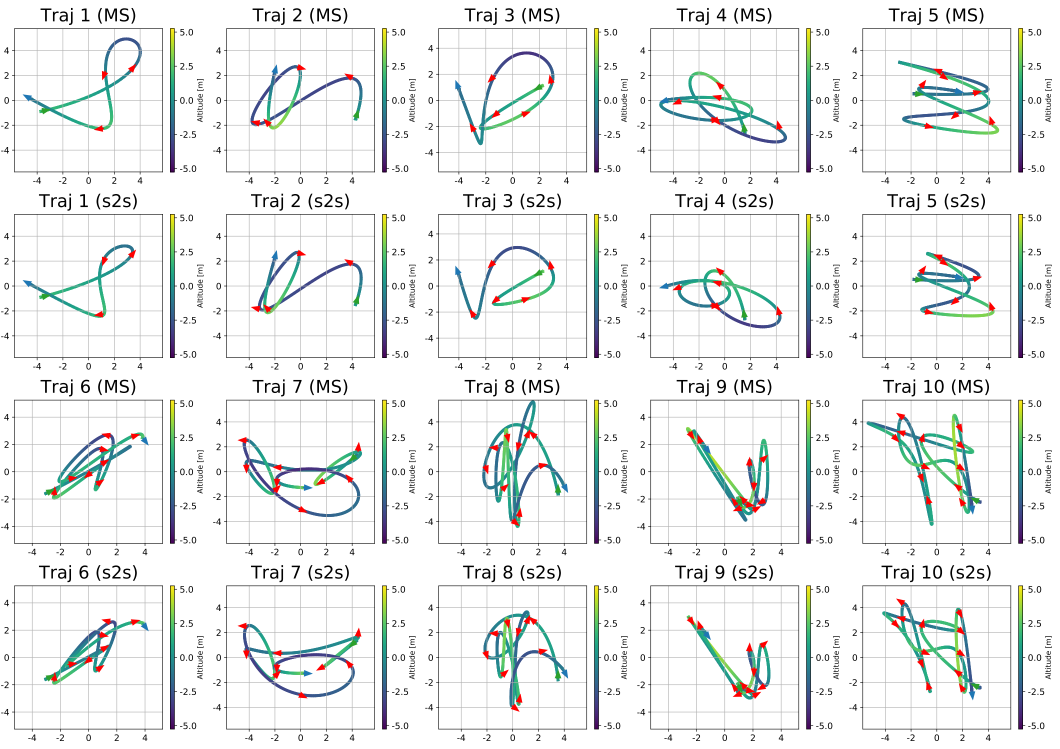


Figure 8: Minimum-snap (MS) and seq2seq (s2s) trajectories evaluated in real-world flight experiments.

Table 5: Length, top speed, and duration of minimum-snap (MS) trajectories with simulation (sim) and real-world (exp) feasibility constraints, as well as trajectory time improvement by seq2seq (s2s) model.

Trajectory			1	2	3	4	5	6	7	8	9	10
Length [m]	MS	sim	26.5	30.5	30.3	44.0	47.3	41.3	52.7	66.1	50.1	70.4
	s2s	sim	22.4	29.2	25.0	35.9	36.4	35.4	49.1	56.3	42.0	59.6
Top spd. [m/s]	MS	sim	6.8	6.5	5.6	6.8	6.5	6.4	6.2	6.5	7.7	6.7
	s2s	sim	7.5	6.7	5.6	5.8	6.4	5.6	5.8	6.4	6.5	6.2
	MS	exp	8.4	8.3	6.2	8.5	6.7	7.0	8.4	7.7	7.8	7.1
	s2s	exp	8.5	8.1	6.0	7.3	5.9	6.3	7.9	7.1	6.6	6.9
Duration [s]	MS	sim	7.2	9.7	9.3	13.2	13.0	12.9	18.4	18.9	16.4	21.5
	s2s	sim	6.4	9.2	8.1	12.0	11.3	11.4	16.9	16.6	15.0	17.4
	MS	exp	5.8	7.7	8.4	10.6	12.7	11.7	13.6	15.8	16.2	20.2
	s2s	exp	5.6	7.6	7.5	9.5	12.2	10.1	12.3	15.0	14.8	15.8
Imp. [%]	s2s	sim	11.9	5.6	13.7	9.4	13.5	11.3	8.1	12.0	8.6	18.7
	s2s	exp	4.3	1.3	10.5	10.4	4.2	14.0	9.2	5.2	8.9	22.1

References

[1] 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. IEEE, 2011.

[2] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.

[3] P. Foehn, A. Romero, and D. Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021.

[4] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen. Optimal time allocation for quadrotor trajectory generation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4715–4722. IEEE, 2018.

[5] A. Romero, R. Penicka, and D. Scaramuzza. Time-optimal online replanning for agile quadrotor flight. *Robotics and Automation Letters (RA-L)*, 2022.

[6] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 2022.

[7] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 59–66. IEEE, 2019.

[8] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza. Learning minimum-time flight in cluttered environments. *Robotics and Automation Letters (RA-L)*, 2022.

[9] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.