Learning Trajectories for Real-Time Optimal Control of Quadrotors

Gao Tang[1], Weidong Sun[2] and Kris Hauser[3]

***Abstract*— Nonlinear optimal control problems are challenging to solve efficiently due to non-convexity. This paper introduces a trajectory optimization approach that achieves real-time performance by combining machine learning to predict optimal trajectories with refinement by quadratic optimization. First, a library of optimal trajectories is calculated offline and used to train a neural network. Online, the neural network predicts a trajectory for a novel initial state and cost function, and this prediction is further optimized by a sparse quadratic programming solver. We apply this approach to a fly-to-target movement problem for an indoor quadrotor. Experiments demonstrate that the technique calculates near-optimal trajectories in a few milliseconds, and generates agile movement that can be tracked more accurately than existing methods.**

I. INTRODUCTION

Nonlinear Optimal Control Problems (OCPs) are critical for high performance in robotics applications. Applications such as Model Predictive Control (MPC) [1] and kinodynamic motion planning [3] require OCPs to be solved quickly and repeatedly. However, OCPs in robotics suffer from nonlinear dynamics, high dimensionality, and non-convex constraints. As a result, they are generally challenging to solve reliably, let alone in real time. In practice, to explore the benefits of trajectory optimization for agile maneuvers, the trajectories are usually computed offline [4] and many approaches [10], [11] are proposed to improve computational efficiency.

We propose a technique that can exploit the versatility of trajectory optimization on handling different dynamics, constraints, and cost functions by using machine learning to help avoid its high computational requirement. There has been an intense interest in using learning to approximately solve OCPs, either using supervised learning [12], [9] or reinforcement learning [13]. We adopt the supervised learning approach and explore the ability of neural networks to perform optimal trajectory prediction. To handle the prediction errors made by the neural network, it is important to introduce some postprocessing in order to respect dynamics and control constraints. We show that only a small amount of optimization suffices to achieve high performance.

Specifically, our technique formulates the trajectory optimization problem as parametric OCP. The range of problems parameters such as initial and final states are defined and the parameters are sampled. A dataset of optimal trajectories corresponding to those sampled parameters are calculated offline (samples shown in Fig. 1). The neural networks are trained using the computed dataset to predict the optimal trajectory for a novel OCP defined by problem parameters as input. While this prediction is ready to be tracked, subsequent one-step refinement based on linearized system dynamics further improves the prediction and is done by solving a sparse quadratic program (QP) within milliseconds.

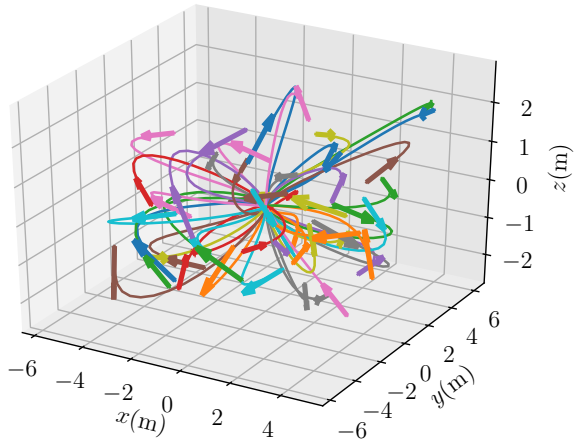

Fig. 1: Samples of a few optimal trajectories in the quadrotor dataset. The arrow shows initial velocity direction.

We evaluate effectiveness of this approach on a quadrotor point-to-point navigation problem. Trained on 50,000 examples, our method calculates near-optimal trajectories in less than 2 ms on average. Experiments demonstrate that trajectories calculated by our technique achieve lower tracking error than minimum-snap trajectories of the same duration [14], [6], [15]. Its fast rate of computation also allows it to be used in a model predictive control (MPC) framework in which the trajectory has to be replanned frequently. Experiments in the supplementary video show the quadrotor using this method to stay above a quickly moving target object.

II. RELATED WORK

Generation of optimal trajectories for dynamic systems in real-time often requires either simplification of system dynamics into linear systems such as double integrator or parameterization of state trajectories in a limited function space such as piecewise polynomials. This paper is concerned primarily with quadrotor trajectory generation, which has been

[1]G. Tang is with department of Mechanical Engineering and Material Science, Duke University, Durham, NC 27708, USA gao.tang@duke.edu

[1]W. Sun is with department of Mechanical Engineering and Material Science, Duke University, Durham, NC 27708, USA ws134@duke.edu

[2]K. Hauser is with the Departments of Electrical and Computer Engineering and of Mechanical Engineering and Materials Science, Duke University, Durham, NC, 27708 USA kris.hauser@duke.edu

explored by many other researchers. Pioneered by Mellinger et al. [14], a quadrotor can explore the differential flatness in its dynamics. The trajectory is parameterized using piecewise polynomials and minimizes a combination of the derivatives of the position states and yaw angle, so-called minimum-snap trajectory. Similar research is found in [6], [15]. These approaches limit the trajectory class to polynomial functions and the available choices of cost functions and constraints are limited. Another drawback is these approaches explore the differential flatness of the quadrotor system and cannot be easily extended to quadrotors augmented with slung loads or arms, or more precise model such as considering air drag.

On the other hand, numerical optimal control is versatile and does not require specific system dynamics, cost function, or constraints. In [7] numerical optimal control is demonstrated on a wide range of quadrotor related trajectory optimization problem. However, it needs to solve non-linear programming (NLP) problems [2] which in general cannot be solved in real time or to a global optimum due to high computational expense.

Machine learning approaches have been proposed to assist in OCP solving by predicting better initial guesses for NLP solvers. Both trajectory optimization [9], [17], [18] and global nonlinear optimization [8] benefits from supervised learning. In [9] precomputed optimal motions are used in a regression to predict trajectories for novel situations to speed up subsequent optimization. This technique works faster than optimizing from scratch, but is not real-time and was not evaluated on dynamic systems. In [17] the nearest-neighbor optimal control (NNOC) method is proposed. This past work only applied to indirect methods for optimal control, which are more challenging to formulate for general problems with state and control constraints. To account for the additional expense of NLP we introduce a faster one-step optimization. Moreover, it uses nearest-neighbor approach for learning, which suffers from the curse of dimensionality.

III. METHODS

Our approach is composed of four major components.

1) Formulate the problem of interest into a parametric OCP.
2) Generate a training database by sampling parameters from a given range and solving for their optimal trajectories.
3) Use a neural network (NN) to learn the mapping from parameters to optimal trajectories.
4) Online, given a new set of problem parameters, use the NN to predict an optimal trajectory, and then solve a one-step QP to refine the prediction.

Although components 2–3 are computationally expensive, they are only performed once offline. Only component 4 is performed repeatedly online, and we demonstrate that it can be performed extremely quickly.

A. Parametric Optimal Control

We address dynamical systems in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

where t is time; $\boldsymbol{x} \in \mathbb{R}^n$ is the state variable; $\boldsymbol{u} \in \mathbb{R}^m$ is the control variable. We refer [14] for the detailed dynamical equations. The state $\boldsymbol{x} = (x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r) \in \mathbb{R}^{12}$ and control $\boldsymbol{u} \in \mathbb{R}^4$. The control is directly chosen as the PWM (scaled to $[0,1]$) for each rotor due to the nonlinear relation between PWM and its thrust and moment [5].

A trajectory is a mapping from time to state and control variables, i.e. $f : t \to [\boldsymbol{x}(t), \boldsymbol{u}(t)], t \in [0, t_f]$. We use direct transcription approach to solve OCPs. An equidistant time grid of size $N+1$, i.e. $\{t_i\}_{i=0}^{N}$ is used for discretization. The trajectory is thus $\boldsymbol{z} = \{t_i, \boldsymbol{x}_i, \boldsymbol{u}_i\}_{i=0}^{N}$. The cost function to be minimized is

$$J = wt_N + h \sum_{i=0}^{N-1} [\boldsymbol{x}_i{}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}_i + \left(\frac{\boldsymbol{u}_i - \boldsymbol{u}_{i-1}}{h} \right)^{\mathrm{T}} \boldsymbol{R} \left(\frac{\boldsymbol{u}_i - \boldsymbol{u}_{i-1}}{h} \right)] \tag{2}$$

where h is the grid size; $\boldsymbol{u}_{-1}$ is the nominal control that compensates gravity; $w, \boldsymbol{Q}, \boldsymbol{R}$ penalizes flight time, state, and change of control variables; the term $(\boldsymbol{u}_i - \boldsymbol{u}_{i-1})/h$ approximates $\dot{\boldsymbol{u}}$. We penalize the change of control since our drone has difficulty ramping up its PWM. The penalty on transfer time w encodes the aggressiveness of the trajectory. This cost function is difficult to be directly optimized using the minimum-snap or related approach. We limit $u \in [0.2, 0.85]$ to avoid saturation when feedback is introduced. Again, control bound on PWM is also difficult to be constrained in the minimum-snap framework. Throughout the paper we use $w \in [0.1, 5]$, $\boldsymbol{Q} = \mathrm{diag}(0,0,0,1,1,1,0,0,0,1,1,1)$, $\boldsymbol{R} = \mathrm{diag}(5,5,5,5)$; System dynamics impose constraints

$$\boldsymbol{x}_{k+1} = \mathrm{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, h) \tag{3}$$

where RK4 means integrating $\boldsymbol{f}$ with constant control $\boldsymbol{u}_k$ for time period h from state $\boldsymbol{x}_k$. We note that RK4 is used for higher integration accuracy and thus N can be reduced. The initial and final states specification imposes constraint

$$\boldsymbol{x}_0 = \boldsymbol{s}_0; \boldsymbol{x}_N = \boldsymbol{s}_f \tag{4}$$

where $\boldsymbol{s}_0$ and $\boldsymbol{s}_f$ are the desired initial and final states. Additionally, depending on specific problem, other path constraint such as collision avoidance and bounds on state and control can be applied.

The problem is to solve the optimal trajectory from a given initial position and velocity (assuming zero angle and angular velocity) to the origin with zero velocity with different choice of aggressiveness encoded by w. We denote the vector collecting 7 problem parameters (3 initial position, 3 initial velocity, and w) as $\boldsymbol{p}$. Due to the limitation of the laboratory size, the initial position is limited within $[-5, -5, -2.5]$ and $[5, 5, 2.5]$ and velocity is limited within $[-2, -2, -1.5]$ and $[2, 2, 1.5]$. Since the initial velocity can be non-zero, this parametric OCP provides the flexibility of commanding the quadrotor to another target in flight without stopping. We only consider the obstacle-free problem, and intend to address obstacles in future work.

B. Learning Optimal Trajectories

The solution to the parametric OCP is a mapping from problem parameter $\boldsymbol{p}$ to the corresponding optimal trajec-

tory $\boldsymbol{z}(\boldsymbol{p})$. This mapping can be approximated using neural networks. The problem parameter is directly treated as a 7-D vector including 3-D position, 3-D velocity, and time penalty weight w. While the position and velocity parameters enables prediction of optimal trajectories based on current state, w controls the aggressiveness of the predicted trajectory. We assume the angle and angular velocity are small and can be controlled much faster than position and velocity so they are not included in problem parameters to simplify the problem. We encode the solution using a long vector, denoted as $\boldsymbol{Z}$ composed of states $\{\boldsymbol{x}_i\}_{i=0}^{N}$, controls $\{\boldsymbol{u}_i\}_{i=0}^{N-1}$, and time t_N. The neural network is simply chosen as a multilayer perceptron (MLP) with one hidden layer. It takes problem parameter as input and the output is the encoded optimal trajectory, i.e. $g(\boldsymbol{w},\boldsymbol{p}):\boldsymbol{p}\rightarrow\boldsymbol{Z}(\boldsymbol{p})$ where $\boldsymbol{w}$ are the weights of the network. Through learning, we find optimal $\boldsymbol{w}$ to minimize

$$L=\mathbb{E}_{p\sim P_{\text{data}}}\text{loss}(g(\boldsymbol{w},\boldsymbol{p}),\boldsymbol{Z}(\boldsymbol{p})) \tag{5}$$

where loss is any regression loss function. We use smoothed L1 loss in this paper.

We train on a dataset of parameter-solution pairs $\{(\boldsymbol{p}^{(1)},\boldsymbol{Z}(\boldsymbol{p}^{(1)})),\ldots,(\boldsymbol{p}^{(M)},\boldsymbol{Z}(\boldsymbol{p}^{(M)})\}$ by sampling a set of problem parameters $\boldsymbol{p}^{(1)},\ldots,\boldsymbol{p}^{(M)}$ and solving their corresponding OCPs using a nonlinear programming (NLP) formulation. To generate the training dataset, both position and velocity are sampled uniformly within range, while w is sampled uniformly after log transformation. We firstly solve 5000 problems to optima using random restart with different initial guesses. The random restart technique increases the probability that the solutions to those problems are indeed globally optimal. These 5000 problem-solution pairs are used as database and the NNOC approach [17], which initializes the local optimizer with several nearest neighbors in the database, is used to solve the rest of the problems. The database can be built incrementally. Eventually we collect $M=50{,}000$ samples. After the whole dataset is built, we resolve all examples again using NNOC to further reduce the likelihood of local optima in the database. Samples of optimal trajectories are shown in Fig. 1.

We train a neural network with input layer of size 7, hidden layer of size 500, and output layer of size 317 (in this paper we choose $N=20$). The hidden layer has a nonlinear activation function, specifically Leaky ReLU with $\alpha=0.2$. 80% data are used for model training and the rest is used as test set. Stochastic gradient descent with momentum is used for training with a mini-batch of 64. The training is terminated when the test error does not decrease within 1000 iterations. Fig. 2 illustrates the learning curves. At the end of training, test error is 1.7×10^{-4}, which indicates that the network approximates the function accurately.

C. Refinement by QP

The neural network is capable of predicting $\boldsymbol{Z}(\boldsymbol{p})$ fairly well for any $\boldsymbol{p}$, but the prediction might not fully respect all the constraints due to approximation error. Although numerical simulation shows the violation is low and the prediction can still be used for trajectory tracking, this prediction

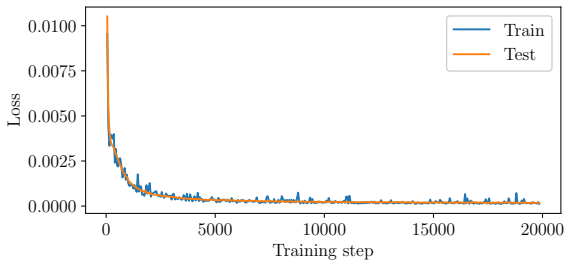

Fig. 2: History of neural net training and test error

improved by subsequent optimization. NLPs are often solved using a sequential quadratic programming (SQP) algorithm, but our refinement technique only solves a QP once. We call this approach one-step QP (OSQP), and demonstrate that it is particularly fast when sparsity is exploited. If more computational resources are available, this approach can be extended to perform a small amount of SQP to further refine the trajectories, e.g., performing backtracking line search or trust region approach and multi-step update.

Assuming the prediction $\boldsymbol{Z}$ is decoded into $\{t_i,\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i\}_{i=0}^{N}$, we want to refine this prediction by finding $\delta\boldsymbol{Z}\equiv\{0,\delta\boldsymbol{x}_i,\delta\boldsymbol{u}_i\}_{i=0}^{N}$ such that $\boldsymbol{Z}+\delta\boldsymbol{Z}$ solves optimal control problem. We note that h is not optimized to keep the problem as QP. As shown later, the prediction error in transfer time t_N is small.

Substituting $\boldsymbol{Z}+\delta\boldsymbol{Z}$ into the cost function and constraints yields

$$\begin{aligned}J=\text{Constant}+h\sum_{i=0}^{N-1}[&\delta\boldsymbol{x}_i{}^{\mathrm{T}}\boldsymbol{Q}\delta\boldsymbol{x}_i+2\bar{\boldsymbol{x}}_i{}^{\mathrm{T}}\boldsymbol{Q}\delta\boldsymbol{x}_i+(\delta\boldsymbol{u}_i\boldsymbol{R}\delta\boldsymbol{u}_i\\&+\delta\boldsymbol{u}_{i-1}\boldsymbol{R}\delta\boldsymbol{u}_{i-1}+2(\bar{\boldsymbol{u}}_i-\bar{\boldsymbol{u}}_{i-1})^{\mathrm{T}}\boldsymbol{R}(\delta\boldsymbol{u}_i-\delta\boldsymbol{u}_{i-1}))/h^2]\end{aligned} \tag{6}$$

which is quadratic in terms of $\delta\boldsymbol{Z}$ and

$$\bar{\boldsymbol{x}}_{k+1}+\delta\boldsymbol{x}_{k+1}=\text{RK4}(\bar{\boldsymbol{x}}_k+\delta\bar{\boldsymbol{x}}_k,\bar{\boldsymbol{u}}_k+\delta\bar{\boldsymbol{u}}_k,h) \tag{7}$$

which is linearized to

$$\bar{\boldsymbol{x}}_{k+1}+\delta\boldsymbol{x}_{k+1}=\bar{\boldsymbol{y}}_k+\frac{\partial\bar{\boldsymbol{y}}_k}{\partial\bar{\boldsymbol{x}}_k}\delta\boldsymbol{x}_k+\frac{\partial\bar{\boldsymbol{y}}_k}{\partial\bar{\boldsymbol{u}}_k}\delta\boldsymbol{u}_k \tag{8}$$

where $\bar{\boldsymbol{y}}_k=\text{RK4}(\bar{\boldsymbol{x}}_k,\bar{\boldsymbol{u}}_k,h)$. Since the neural network predicts $\boldsymbol{Z}$ close to the optimum, $\delta\boldsymbol{Z}$ is small and linearization is valid. We conveniently change the nonlinear dynamics constraints into linear constraints. Additionally, the constraints for initial and final states are readily converted into

$$\bar{\boldsymbol{x}}_0+\delta\boldsymbol{x}_0=\boldsymbol{s}_0;\bar{\boldsymbol{x}}_N+\delta\boldsymbol{x}_N=\boldsymbol{s}_f \tag{9}$$

and other constraints such as bounds on state and control variables can be converted similarly.

We note that the neural network predicts a trajectory with zero angle and angular velocity which might be different from the quadcopter's current state. This issue is further reduced by solving optimal $\delta\boldsymbol{Z}$ since to satisfy Eq. (9) the result trajectory has an initial state identical to the quadcopter's current state.

These constraints are assembled into a QP of the form

$$\begin{aligned} \underset{x}{\text{Minimize}} \quad & \frac{1}{2} x^{\mathrm{T}} P x + q^{\mathrm{T}} x \\ \text{subject to} \quad & l \leq Ax \leq u \end{aligned} \tag{10}$$

with positive semi-definite matrix P. From Eq. (9) $\boldsymbol{\delta x}_0$ and $\boldsymbol{\delta x}_N$ can be determined directly. The optimization variables for QP are thus $\{\boldsymbol{\delta x}_i\}_{i=1}^{N-1}$ and $\{\boldsymbol{\delta u}_i\}_{i=0}^{N-1}$. Observing that $\boldsymbol{Q}$ and $\boldsymbol{R}$ are both diagonal in Eq. (6), P is also diagonal. The linear constraints contain linear equality constraints from Eq. (8) and inequality constraints of bounds on state and control. The bounds on state and control variables are essentially block identity matrix in A. There are N sets of linearized dynamical constraints as Eq. (8) for $k = 0, \ldots, N-1$. Each instance of (8) introduces at most $n \times (n+m+1)$ nonzero elements. Out of those nonzero elements, n^2 belong to $\frac{\partial \bar{\boldsymbol{y}}_k}{\partial \bar{\boldsymbol{x}}_k}$, nm belong to $\frac{\partial \bar{\boldsymbol{y}}_k}{\partial \bar{\boldsymbol{u}}_k}$, and the rest n is the diagonal matrix associated with $\boldsymbol{\delta x}_{k+1}$. To exploit this sparsity, we use the implementation in [16], and all tested problems can be solved in microseconds.

IV. RESULTS

In this section, we evaluate the method's performance in simulation and on a real quadcopter.

A. System Description

We use a commercially available quadrotor Crazyflie 2.0[1], with basic specifications listed in Tab. I. We refer to [5] for more details on system dynamics. The position of the quadrotor is captured by the Vicon motion capture system and transmitted to ground control station using Ethernet at 200 Hz. The raw data stream from Vicon goes through a Kalman filter and then serves as feedback for a position controller on the ground control station. The position controller is a proportional-integral-differential (PID) controller, and sends commands at 100 Hz through radio to the quadrotor which drives the on-board attitude controller at 500 Hz. The commanded target position is fed into the neural network to generate a trajectory and then optimized by OSQP to get the optimal trajectory. The optimal trajectory is then sent to the position controller as a reference trajectory. The system diagram is shown in Fig. 3.

TABLE I: Specifications of Crazyflie2.0

Parameter	Value
Mass (with markers)	33.5 g
Size(W×H×D)	92×29×92 mm
Takeoff weight	42 g
Flight time	7 min

[1]https://www.bitcraze.io/crazyflie-2/

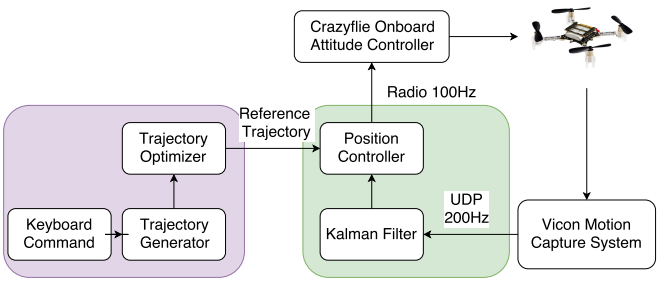

Fig. 3: Architecture of the system

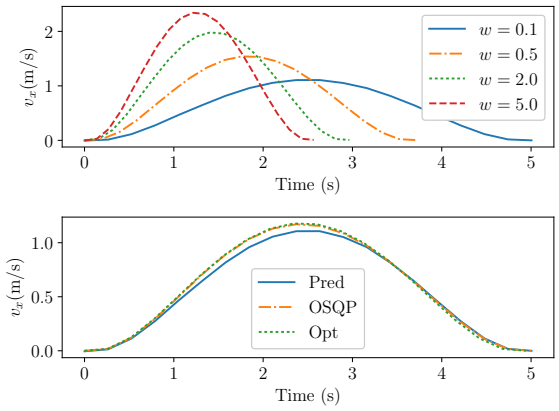

Fig. 4: Top: four trajectories from the same initial state with different aggressiveness weights. Bottom: predicted, OSQP refined, and optimal trajectories for $w = 1$. The curves almost overlap, indicating high prediction accuracy.

B. Numerical Validation

The top row of Fig. 4 shows the prediction of optimal trajectories from $(-3, -3, -2)$ with zero velocity to the origin with different weights on transfer time. These show that the chosen aggressiveness affects the optimized transfer time. The bottom row indicates that the neural network makes accurate predictions. Refinement by QP is able to further optimize the trajectory, and the result is visually indistinguishable from the global optimum.

We evaluate the network's prediction error in transfer time in Fig. 5 and Tab. II. It shows that most of the errors are within 0.25 s. This is important because OSQP is unable to improve the transfer time. However, there are still a few outliers with large transfer time error. These tend to be non-aggressive problems. For example, in the worst problem, $w = 0.15$ and the costs from the OSQP and optimal trajectory are 0.870 and 0.739, respectively.

Next, we examine the violation of the dynamics constraint.

TABLE II: Prediction error in transfer time (s)

MAE	RMSE	max	median
0.056	0.080	1.62	0.043

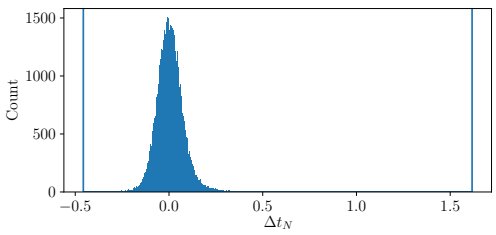

Fig. 5: Prediction error in transfer time. Most prediction errors are within 0.25 s. Outliers (invisible in histogram) are indicated with vertical lines.

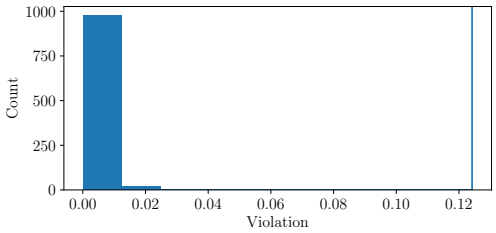

Fig. 6: Histogram of constraint violation measured by the norm of the constraint function. Outliers (invisible in histogram) are indicated with vertical lines.

We randomly sample 1000 initial states and evaluate the L2 norm of the violation of constraints of the optimal trajectories from our approach (0 means no violation). The result is shown in Fig. 6 with a worst case of 0.12. Considering that this problem has $(N-1)n = 228$ constraints, even the worst-case violation is relatively small and can be well compensated by feedback control.

Tab. III compares the running time and cost of the following methods:

1) Our approach (NN+OSQP)
2) Minimum-snap trajectory (Min-Snap) [14]
3) NLP solver with straight line initialization (SL+NLP)
4) NLP solver with NNOC initialization (NNOC) [17]
5) NLP solver with neural net initialization (NN+NLP)

on 1000 sampled initial states. We note that Min-Snap is implemented in Python and a careful C++ implementation can reduce the computation time to the same level with our approach. The transfer time of minimum-snap trajectory must be set by some other methods which often leads to conservativeness for safety. In these experiments it is selected to be the same with the results from NNOC. Since the minimum-snap approach is optimizing the snap of selected state variable, it will have larger cost for our cost function.

Compared with the full NLP solver, our approach is able to get an approximate solution *two orders of magnitude* faster at similar levels of cost. It also obtains better cost than Min-Snap. Although we are not explicitly optimizing control energy, it turns out our approach yields lower energy trajectory. Besides, Min-Snap has to check violation of control constraint a posterior and as a result, the transfer time has to be chosen conservatively in practice.

TABLE III: Comparison between approaches.

	NN+OSQP	Min-Snap	SL+NLP	NNOC	NN+NLP
Success	1000	1000	563	1000	1000
Time (ms)	1.80	10.21	382.9	194.7	131.1
Avg. cost [2]	8.73	8.88	8.64	8.64	8.64
Avg. energy	6.67	7.00	6.69	6.69	6.69
Avg. jerk	0.37	0.40	0.35	0.35	0.35

C. Point-to-point Navigation and Real-time Tracking

Fig. 7 compares our method applied to the real quadcopter by a point-to-point maneuver from (0, 0, 0) to (3, 3, 1.5). Our approach predicts a transfer time of 3.1 s. The minimum-snap trajectory to reach the same target within the same amount of time is also shown. The two reference trajectories are quite different, especially in the z direction. This is not too surprising because they are optimizing different cost functions. The trajectory from our prediction yields better tracking performance. For underpowered drone like Crazyflie, the specialized cost function in Eq. (2) penalizes rapid change of rotor PWM so it leads to better performance than the general minimum-snap approach.

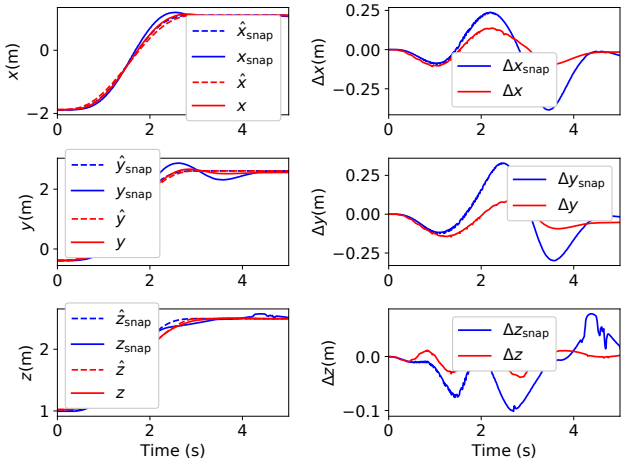

Fig. 7: Results for tracking a trajectory. The dashed and solid lines are reference and actual trajectory. The blue and red lines are the trajectory by our minimum-snap and our approach. To reach the same target within the same amount of time, our approach generates a trajectory with better tracking performance.

We repeated this experiment for 10 manually selected targets. The tracking errors, measure by the norm of position and velocity errors, are listed in Tab. IV. It shows our approach generates a trajectory that is easier to track than the minimum-snap approach, given the same transfer time.

Since our approach can predict a trajectory with non-zero initial velocity, it can switch target rapidly. Fig. 8 demonstrates this capability. A movement to the initial target is interrupted with a new target after 1.7 s. The technique smoothly and immediately switches to the new trajectory.

[2] See Eq. (2)

TABLE IV: Comparison of average tracking error

NN+OSQP	Min-Snap
0.45	0.87

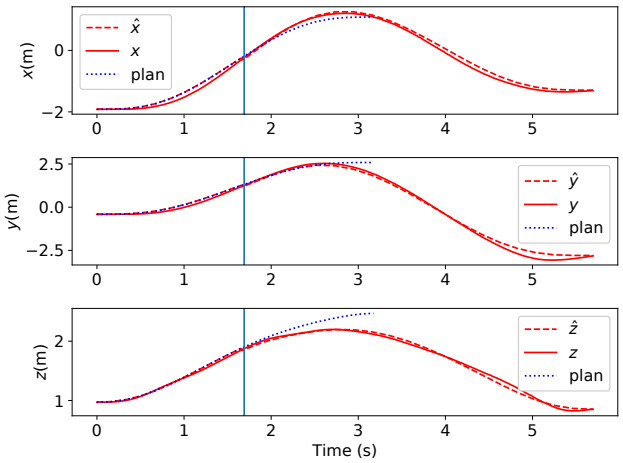

Fig. 8: Results for replanning during tracking. The vertical line indicates when replanning is commanded. The blue curve shows the planned trajectory to the first target. Our approach is able to generate optimal trajectory in real time.

A supplementary video shows another real-time tracking experiment. Here, we set the quadrotor to fly above a moving target moved by a human at a certain height, as shown in Fig. 9. Our approach enables agile response of the quadrotor when the target is moved to a large distance.

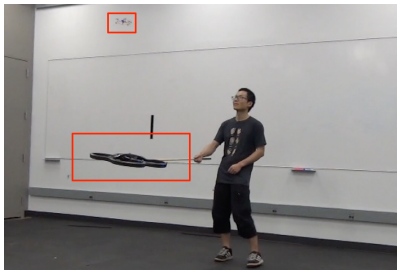

Fig. 9: One frame of the live experiment. The rectangles show the drone and target.

V. CONCLUSION

We exploit the ability of machine learning for global nonlinear function approximation and efficiency of local trajectory optimizer to enable real-time OCP solving.

The problem of interest is formulated as parametric OCP so dataset of optimal trajectories is created and the parameter-solution mapping can be learned. It turns out the optimal trajectories can be learned using small amount of data and approximated to high precision. The local trajectory optimizer based on sparse QP benefits from the high accuracy of the prediction from the learned model. The combination of these techniques enables real-time solving of challenging nonlinear OCPs. We validate this approach using an indoor quadrotor system. We note that this quadrotor is underpowered and light-weight so its disturbance rejection capabilities is relatively weak. In future work we intend to apply our technique to quadrotors capable of more aggressive maneuvers. Our method should benefit more from the exploitation of nonlinear dynamics to achieve higher performance. Future work includes applying this technique to more challenging systems such as locomotion and theoretical study of stability guarantee and bounds on loss of cost function.

REFERENCES

[1] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proc. American Control Conf.*, vol. 1–6, 2000, pp. 872 – 876.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[3] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[4] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload," in *Robotics: Science and Systems*, 2017, pp. 1–10.

[5] J. Förster, M. Hamer, and R. D'Andrea, "System identification of the crazyflie 2.0 nano quadrocopter," B.S. thesis, 2015.

[6] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2016, pp. 139–146.

[7] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2958–2964.

[8] K. Hauser, "Learning the problem-optimum map: Analysis and application to global optimization in robotics," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 141–152, Feb. 2017.

[9] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 1-2, pp. 111–127, Jan. 2013.

[10] F. Jiang and G. Tang, "Systematic low-thrust trajectory optimization for a multi-rendezvous mission using adjoint scaling," *Astrophysics and Space Science*, vol. 361, no. 4, p. 117, 2016.

[11] F. Jiang, G. Tang, and J. Li, "Improving low-thrust trajectory optimization by adjoint estimation with shape-based path," *Journal of Guidance, Control, and Dynamics*, pp. 1–8, 2017.

[12] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3719–3726.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

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

[15] M. W. Mueller, M. Hehn, and R. D'Andrea, "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310.

[16] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *arXiv preprint arXiv:1711.08013*, 2017.

[17] G. Tang and K. Hauser, "A data-driven indirect method for nonlinear optimal control," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. –.

[18] T. Tomić, M. Maier, and S. Haddadin, "Learning quadrotor maneuvers from optimal control and generalizing in real-time," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1747–1754.