2017 11th Asian Control Conference (ASCC)
Gold Coast Convention Centre, Australia
December 17-20, 2017

Towards the Realtime Sampling-based Kinodynamic Planning for Quadcopters

Menglu Lan[1,*], Shupeng Lai[2,*] and Ben M. Chen[2]

Abstract— **This paper presents a sampling-based kinodynamic planning algorithm for quadcopters which have a high dimensional state space amid state constraints. The proposed approach utilizes BIT* as the main planner to perform the ordered search directly in state space, without decoupling the geometric and dynamic constraints. Instead of using a general BVP solver for non-trivial local steering problem, a neural network based approximation is adopted to boost the computational efficiency. The potential of the proposed approach towards a real-time onboard implementation is demonstrated by various simulations and a real flight test.**

I. Introduction

In recent years, quadcopters received significant research attention for their light-weight, exceptional agility, relatively simple control structure, and huge potentials for aerial surveillance, reconnaissance, and inspection in complex and dangerous environment. Like many other mobile robots, the *motion planning* of the quadcopters, i.e., finding a collision-free and dynamically feasible trajectory between initial and end position, is an indispensable part of the whole navigation system. However, there are long-standing challenges in designing computationally-efficient motion planning algorithm that can handle complex systems with high dimensional space, particularly for realistic system involved with non-holonomic constraints or kinodynamic constraints. Quadcopter is a classical example of such a "difficult" system with a high-dimensional state space. Due to the actuator limitation, it also imposes extra state constraints such as limited velocity and acceleration. Moreover, for micro-sized quadcopters, the on-board computational resource is generally limited due to its constrained size and payload. Such limited resources have to be shared with other computational intensive modules such as SLAMs, which makes the real-time on-board motion planning even challenging, especially in cluttered unknown environment.

Traditionally, to address the mentioned motion planning problem, the original complex problem is decoupled into two layers for better computational efficiency. A geometric global planner is firstly adopted to compute a geometric collision-free path or corridor without considering any vehicle dynamics. Then a dynamically-feasible trajectory is generated along the path or within the corridor. Such a decoupling method is computationally efficient and demonstrates a fairly good performance [4], [5]. However, the two-layered structure has an inherited drawback that the trajectory layer may find solution with poor quality or fail to find any feasible solution. This is due to the ignored system dynamics in the first layer which causes the geometric planner to produce unreasonable geometric path.

The emergence of sampling-based planners such as RRT makes the directly solving of motion planning problem for dynamical system possible. These methods randomly grow a tree in state space by sampling a control input and propagate the state, which finds the feasible solution much quicker compared to graph-search-based methods [6]. However, RRT itself is provably non-optimal and its optimal variants such as RRT* normally require a steering function between two states. To apply these optimal planners into dynamic systems, a two-point-boundary-value problem (TPBVP) solver is often adopted to steer the state [12]. Though there are much advances in the TPBVP solvers and many of them can manage to solve one trajectory within reasonable time, the planner still suffers from the low convergence rate due to the numerous calls to the steering function. To reduce the number of online calls to the TPBVP solver, [13] has built up a TPBVP solution look-up table at off-line from a set of pre-sampled nodes, and added back the initial state and end state to the pre-constructed "roadmap" during the online process. However, the solution quality can be affected by the number of pre-sampled nodes.

This paper proposed an innovative way to directly solve the complex motion planning problem for quadcopters without decoupling the collision checking and vehicle dynamics. It utilizes a sampling-based planner named Batch Informed Tree (BIT*) [10], which is a heuristic guided ordered-search variant of the original RRT* but has much faster convergence rate. Like any other tree planners, BIT* incrementally builds a tree rooted from start state (vertex) towards the goal state (vertex). Any edge (i.e., state to state motion) found to be in collision would be discarded during the growing process. However, instead of directly using a time-consuming TPBVP solver online as its steering function, the proposed approach utilizes a neural-network based approximation to boost the convergence rate. Like in [13], the main rational behind is to bring any possible computation offline and thus release the online computational burden. For each pair of states and corresponding state trajectory, the minimum cost of the trajectory, the position profile as well as the optimal time duration of the trajectory is trained and approximated by a multilayer perceptron (MLP) network. The approximated

[1]The author is with NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore {lanmenglu}@u.nus.edu

[2]The authors are with the Department of Electrical & Computer Engineering, National University of Singapore, Singapore {elelais, bmchen}@nus.edu.sg

*These two authors contributed equally.

978-1-5090-1573-3/17/$31.00 ©2017 IEEE

cost is used for the ordered search and approximated position profile is for collision checking, and the time duration is used to recover the full state trajectory along the tree vertices found by BIT*. The major difference of our approach compared to [13] is that we do not limit the search within a set of the pre-sampled nodes. Also, since we adopt a more complex dynamic model, no further smoothing of the generated trajectory is required.

The paper is organized as follows. Sec. II presents the formal problem statement of optimal motion planning problem. Sec. III discusses the dynamic model for quadcopter used in this paper. In Sec. IV, a formal formulation for local steering is provided. The details of neural network training is provided in Sec. V. Simulation and experiment results, including a real flight test, are demonstrated in Sec. VI. Finally, conclusion and remarks are drawn in Sec. VII.

II. Problem Statement

Let $\mathcal{X} \subseteq \mathbb{R}^n$ represent the state space and $\mathcal{U} \subseteq \mathbb{R}^m$ represent the control space of the vehicle respectively, with the system dynamics described by its state space model:

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

where t denotes the time, $\mathbf{x}(t) \in \mathcal{X}$ is a state of the vehicle and $\mathbf{u}(t) \in \mathcal{U}$ is a control input of the vehicle. Let $\mathcal{X}_{obs} \subset \mathcal{X}$ and $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ be the obstacle region and free region of the state space respectively, and $\mathcal{U}_{free} \subset \mathcal{U}$ be the free control space in which the control inputs satisfy the input constraints given by the users. The optimal kinodynamic motion planning then can be formally stated as follows.

Given a start state $\mathbf{x}_{start} \in \mathcal{X}_{free}$ and a goal state $\mathbf{x}_{goal} \in \mathcal{X}_{goal} \subset \mathcal{X}_{free}$, find a *trajectory* $\pi =< T, \mathbf{x}, \mathbf{u} >$ such that applying control $\mathbf{u} : [0, T] \to \mathcal{U}_{free}$ for some time duration T, the resulted state trajectory $\mathbf{x} : [0, T] \to \mathcal{X}_{free}$ between $\mathbf{x}(0) = \mathbf{x}_{start}$ and goal state $\mathbf{x}(T) = \mathbf{x}_{goal}$ minimizes the *cost* or *performance index*:

$$J(\pi) = \phi(\mathbf{x}(T), T) + \int_0^T \mathbf{L}(\mathbf{x}(t), \mathbf{u}(t))\mathrm{d}t, \tag{2}$$

where ϕ and $\mathbf{L}$ are some arbitrary weighting functions. Then the optimal collision-free trajectory π^*_{free} is

$$\begin{aligned}\pi^*_{free} = \operatorname{argmin}\{\pi | \mathbf{x}(0) = \mathbf{x}_{start} \wedge \mathbf{x}(T) = \mathbf{x}_{goal} \wedge \\ \forall\{t \in [0, T]\}(\mathbf{x}(t) \in \mathcal{X}_{free} \wedge \mathbf{u}(t) \in \mathcal{U}_{free})\} J(\pi)\end{aligned} \tag{3}$$

In this paper, we choose a performance index in a form of

$$J(\pi) = \int_0^T m + \mathbf{u}(t)^\intercal R \mathbf{u}(t) dt, \tag{4}$$

where m is a weighting factor of time and R is a positive-definite weighting matrix for control inputs.

III. Planning Model

The model of a quadcopter is highly non-linear and has high degrees of freedom. Direct planning over its full model is difficult and slow. However, the quadcopter has been proven as a deferentially flat system [1] with the flat output as its position x, y, z and yaw ϕ. Therefore, it is a common practice to use a simplified, integrator-based model during the trajectory planning stage, such as the double integrator used in [2], the third order model used in [3], [4] and the quadruple integrator model used in [1], [5]. In this paper, a third order model similar to [3] is adopted. In our previous study of generating smooth trajectories for quadcopter [7], this model shows satisfying results by producing a minimum jerk solution.

The control inputs to this third order model are the rotational rates about the vehicle body axis ω_x, ω_y, ω_z and the mass normalized collective thrust F. The vehicle's motion can be described as

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = {}^o_b\mathrm{R} \begin{bmatrix}0\\0\\F\end{bmatrix} + \begin{bmatrix}0\\0\\-g\end{bmatrix} \tag{5}$$

where g is gravitational acceleration. And the matrix ${}^b_o\mathrm{R}$ is the rotational matrix from the vehicle's body frame $\mathbb{B}$ to the inertial frame $\mathbb{O}$. The vehicle's attitude is related to the rotational rates as

$${}^o_b\dot{\mathrm{R}} = {}^o_b\mathrm{R} \begin{bmatrix}0 & -\omega_z & \omega_y\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix} \tag{6}$$

For later convenience, we also define the mass-normalized global force $H \in \mathbb{R}^3$ as

$$H := \begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} + \begin{bmatrix}0\\0\\g\end{bmatrix} = {}^o_b\mathrm{R} \begin{bmatrix}0\\0\\F\end{bmatrix} \tag{7}$$

Our design target is to generate a smooth trajectory that cause less wobbling in the vehicle's attitude. This helps preventing pendulum effect when the vehicle carries payload that increasing its moment of inertia, such as a heavy calligraphy brush shown in [7]. The design target is equivalent as penalizing the rotational speed about the vehicle body axis $\|\omega_x\|$, $\|\omega_y\|$ and $\|\omega_z\|$. Since it is possible to fix the yaw angle of the vehicle, resulting no rotation around its body z axis, we focus on minimizing $\|\omega_x\|$ and $\|\omega_y\|$. Furthermore, the global force H shall also be limited due to the rotors' maximum output as

$$\begin{aligned}&\|H\| = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{x} + g)^2} > H_{\min}\\ &\|H\| < H_{\max}\end{aligned} \tag{8}$$

It is shown in [3] that the constraints in Eqn. 8 can be satisfied by limiting

$$|\ddot{x}| \le \ddot{x}_{\max},\ |\ddot{y}| \le \ddot{y}_{\max},\ |\ddot{z}| \le \ddot{z}_{\max} \tag{9}$$

whereas the upper bound of $\|\omega_x\|$ and $\|\omega_y\|$ can be expressed as

$$\|\omega_{x,y}\| \le \frac{\left\|\dot{H}\right\|}{\|H\|}. \tag{10}$$

Note the denominator $\|H\|$ is already constrained by Eqn. 8. Thus we could minimize $\left\|\dot{H}\right\|$ for penalizing $\|\omega_x\|$ and

$\|\omega_y\|$ indirectly. From Eqn. 7, the minimization of $\left\|\dot{H}\right\|$ can be achieved by minimizing

$$\int (\dddot{x}^2 + \dddot{y}^2 + \dddot{z}^2)dt \tag{11}$$

over the whole trajectory. We notice that the Eqn. 11 is a part of Eqn. 4 with $\mathbf{u}(t)$ being the jerk of the system.

IV. Optimization problem for local steering

In this section, we present the formulation for the local steering problem of the BIT* algorithm. For each pair of states, summarizing the objectives and constraints in Sec. II and Sec. III, the following non-linear optimization problem can be formulated

$$\begin{aligned}
&\min_{\mathbf{j}(t),T,t\in[0,T]} \left\{ J = \textstyle\int_{t=0}^{T} m + \mathbf{j}(t)^2 dt \right\} \ s.t. \\
&\mathbf{p}(0) = \mathbf{p}_{ini}, \quad \mathbf{p}(T) = \mathbf{p}_{\text{ref}} \\
&\mathbf{v}(0) = \mathbf{v}_{ini}, \quad \mathbf{v}(T) = \mathbf{v}_{\text{ref}} \\
&\mathbf{a}(0) = \mathbf{a}_{ini}, \quad \mathbf{a}(T) = \mathbf{a}_{\text{ref}} \\
&\dot{\mathbf{p}}(t) = \mathbf{v}(t) \\
&\dot{\mathbf{v}}(t) = \mathbf{a}(t) \\
&\dot{\mathbf{a}}(t) = \mathbf{j}(t) \\
&-\mathbf{v}_{\max} \le \mathbf{v}(t) \le \mathbf{v}_{\max}, \quad \forall t \in [0, T] \\
&-\mathbf{a}_{\max} \le \mathbf{a}(t) \le \mathbf{a}_{\max}, \quad \forall t \in [0, T]
\end{aligned} \tag{12}$$

where T is the total time of the trajectory, m is a weighting factor, $\mathbf{j}$, $\mathbf{a}$, $\mathbf{v}$, $\mathbf{p}$ are the jerk, acceleration, velocity and position of the trajectory respectively.

If we denote $\mathbf{p}_k$, $\mathbf{v}_k$, $\mathbf{a}_k$, $\mathbf{j}_k \in \mathbb{R}^{K\times 3}$ as the 3 dimensional position, velocity, acceleration and jerk of the vehicle at discrete times $k \in 1, ..., K$, the discrete trajectory then follows

$$\begin{aligned}
\mathbf{a}_{k+1} &= \mathbf{a}_k + h\mathbf{j}_k \\
\mathbf{v}_{k+1} &= \mathbf{v}_k + h\mathbf{a}_k + \tfrac{h^2}{2}\mathbf{j}_k \\
\mathbf{p}_{k+1} &= \mathbf{p}_k + h\mathbf{v}_k + \tfrac{h^2}{2}\mathbf{a}_k + \tfrac{h^3}{6}\mathbf{j}_k
\end{aligned} \tag{13}$$

where h is the discretization time step. The similar formulation for quadcopter trajectory generation can be found in [2]. From Eqn. 13, the acceleration, velocity and position of the trajectory can be expressed as affine functions of the jerk as

$$\begin{aligned}
\mathbf{a}_{k+1} &= \mathbf{a}_0 + h(\mathbf{j}_0 + \mathbf{j}_1 + ... + \mathbf{j}_k) \\
\mathbf{v}_{k+1} &= \mathbf{v}_0 + (k+1)h\mathbf{a}_0 + \tfrac{h^2}{2}[(2k-1)\mathbf{j}_0 \\
&\quad + (2k-3)\mathbf{j}_1 + ... + \mathbf{j}_k] \\
\mathbf{p}_{k+1} &= \mathbf{p}_0 + (k+1)h\mathbf{v}_0 + \tfrac{(k+1)^2h^2}{2}\mathbf{a}_0 \\
&\quad + \tfrac{h^3}{6}\{[3(k+1)k+1]\mathbf{j}_0 \\
&\quad + [3k(k-1)+1]\mathbf{j}_1 + ... + \mathbf{j}_{k-1}\}
\end{aligned} \tag{14}$$

Eqn. 14 can be expressed in matrix from, giving linear relationship from jerk to acceleration, velocity and position, which can later be used to express boundary condition and state constraints.

The discrete version of the problem in Eqn. 12 can now be written as

$$\begin{aligned}
&\min_{\mathbf{j}_k,h,k\in[0,K-1]} \left\{ J = \textstyle\sum_{k=0}^{K-1} \mathbf{j}_k^2 + mKh \right\} \ s.t. \\
&\mathbf{p}_0 = \mathbf{p}_{ini},\ \mathbf{p}_K = \mathbf{p}_{\text{ref}} \\
&\mathbf{v}_0 = \mathbf{v}_{ini},\ \mathbf{v}_K = \mathbf{v}_{\text{ref}} \\
&\mathbf{a}_0 = \mathbf{a}_{ini},\ \mathbf{a}_K = \mathbf{a}_{\text{ref}} \\
&\mathbf{a}_{k+1} = \mathbf{a}_k + h\mathbf{j}_k \\
&\mathbf{v}_{k+1} = \mathbf{v}_k + h\mathbf{a}_k + \tfrac{h^2}{2}\mathbf{j}_k \\
&\mathbf{p}_{k+1} = \mathbf{p}_k + h\mathbf{v}_k + \tfrac{h^2}{2}\mathbf{a}_k + \tfrac{h^3}{6}\mathbf{j}_k \\
&-\mathbf{v}_{\max} \le \mathbf{v}_k \le \mathbf{v}_{\max}, \quad \forall k \in [1, K] \\
&-\mathbf{a}_{\max} \le \mathbf{a}_k \le \mathbf{a}_{\max}, \quad \forall k \in [1, K]
\end{aligned} \tag{15}$$

The number of discrete steps K is a preset constant whereas the duration of each step h is a programming variable that determines the total duration of the trajectory as Kh. And with Eqn. 14, all the boundary condition and state constraints can be written as

$$\begin{aligned}
A_{\text{eq}}(h)\hat{j} &= b_{\text{eq}}(h) \\
A(h)\hat{j} &\le b(h)
\end{aligned} \tag{16}$$

where $\hat{j} \in \mathbb{R}^{3K\times 1}$ is obtained by cascading the vectors of each dimension of $\mathbf{j} \in \mathbb{R}^{K\times 3}$.

Sequential quadratic programming (SQP) is commonly used to locally optimize the non-convex, constrained problem in Eqn. 15. It optimize all the programming variables $\mathbf{j}_k$ and h simultaneously and works by repeatedly solve quadratic approximation of the original problem over all the programming variables. However, we found that once the step duration h in Eqn. 15 is fixed, its objective function becomes quadratic

$$\min_{\mathbf{j}_k,k\in[0,K-1]} \left\{ J = \sum_{k=0}^{K-1} \mathbf{j}_k^2 \right\} \tag{17}$$

with all of its constraints in Eqn. 16 linearized with the fixed h. The new problem is now both convex and quadratic with a unique minimum cost upon solution. In summary, for a given h, we can transform the problem in Eqn. 15 to a quadratic one with a unique minimum cost which can be written as

$$J_{\text{c}} = \bar{f}(h). \tag{18}$$

The optimization problem in Eqn. 15 now becomes

$$\min_{h} \left\{ \bar{f}(h) \right\} \tag{19}$$

which is a single variable optimization problem. It is then solved through a binary search over the value of h. Similar to SQP, the core of this approach is still to solve a series of quadratic problem resulted by fixing h. However, unlike the general SQP approach, it guarantees a feasible solution even if the bisection approach is terminated prematurely. Due to the non-convex nature of the original problem, there is no guarantee that both algorithm will converge to the same global optimum. But to our experience in solving smooth quadrotor trajectories, they turns to give the same solution upon convergence.

V. Neural network training

Towards the real-time implementation of the sampling based kinodynamic planner, we follow the idea to bring all possible calculations offline. We identify that for the BIT* algorithm, three pieces of information related the local steering (see Eqn. 15) are important.

- The position profile of the trajectory.
- The minimum cost of the trajectory.
- The optimal time step length h.

The position profile of the trajectory is for the geometric collision checking process. The minimum cost is necessary for calculating the cost-to-come for each state and aids the BIT* to perform the ordered search together with a heuristic function. Finally, the optimal time step length h is preferred, so that the non-convex optimization in Eqn. 15 can be transferred into a convex and quadratic one. This transformation is illuminated during our bisection search approach in Sec. IV.

It is generally expensive to solve a non-convex, constraint optimization problem to acquire these three pieces of information. Hence we propose a neural network based approach, that utilizes the offline trained neural networks to check for these information during online execution. Three multilyater perception (MLP) networks are constructed to acquire each of these three pieces of information. The MLPs are designed to handle the BVP with a preset value of $K = 25$, $m = 12$, $\mathbf{v}_{\max} = [1.5,\ 1.5,\ 1]$, $\mathbf{a}_{\max} = [1,\ 1,\ 0.5]$ and arbitrary initial and final states. To reduce the dimension of the MLP's input vector, it is assumed that all trajectory is started at initial position $[0,\ 0,\ 0]$ so that we could remove the initial position from the input. During online usage, the resulted trajectory is shifted to any starting point by adding the actual initial position back. Thus the three MLPs share the same input $\mathbf{L} = [\mathbf{v}_{ini}^T,\ \mathbf{a}_{ini}^T,\ \mathbf{p}_{ref}^T,\ \mathbf{v}_{ref}^T,\ \mathbf{a}_{ref}^T]^T \in \mathbb{R}^{15\times 1}$. Whereas the output for each MLP differs based on the information needed.

For the shape of the trajectory, the output is a series of line segment described by discrete 3 dimensional points to approximate the original trajectory. This approximation is a common practice for online collision detection as it can deal with any type of trajectories and environmental information expressed in occupancy or cost map.

The labels are acquired by uniformly sampling the resulted trajectory's position of a solved BVP. In our case, a total of 10 line segments is adopted. Since the trajectory always start from position $[0,\ 0,\ 0]$, the first position point is omitted, resulting a output vector $\mathbf{P}_{\mathrm{t}} \in \mathbb{R}^{27\times 1}$. For the trajectory's cost and optimal time step, the output is a single scalar value and the labels are also found through the solving of BVPs. During the MLP designing process, a deep MLP with 4 layers and each layer with 2048 neurons is first adopted to over fit the training data. Then it is gradually reduced to a smaller size for faster evaluation. The structure of reduced networks can be found in Table. I.

All the input and hidden layers use the rectified linear unit (ReLU) activation function. The ReLU function allows to evaluate the neural network as matrix multiplication. It helps to efficiently implement the MLP on devices without GPU using linear algebra packages such as Eigen[8].

TABLE I: Nerual Network Structure

	Path	Cost	Time step
Input	1024 (ReLU)	1024 (ReLU)	512 (ReLU)
Hidden 1	1024 (ReLU)	1024 (ReLU)	512 (ReLU)
Hidden 2	1024 (ReLU)	512 (ReLU)	512 (ReLU)
Hidden 3	512 (ReLU)	-	-
Output	27 (Linear)	1 (Linear)	1 (Linear)

TABLE II: Network performance

	Path	Cost	Time step
Training MSE	0.0147	0.0705	4.4142e-06
Testing MSE	0.0305	0.45	9.344e-06

During the training process, the amount of training data is gradually increased until the mean squared error (MSE) of both the training and testing set is small enough. Finally, a total of 300,000 training data is utilized with a testing set of 30,000 data. The performance of the trained MLPs are given in Table. II. The efficiency of the trained neural network is compared to the TPBVP solver which generates all the training data. The comparison is done in a Matlab environment, with a laptop equipped with an Intel I7 7500U CPU (without the usage of GPU). The average solving time of the presented bisection TPBVP solver is 171 ms while the average time consumed for evaluating all three neural networks is 1.7 ms. In Fig. 1, a pair of trajectories with the same boundary conditions are compared. The solid line shows the actual trajectory solved by the BVP solver, while the dot triangle line shows the trajectory predicted by the path and time networks.

VI. Simulation and Experimental results

The proposed method is tested by various simulations and a real flight test, which are detailed in Sec. VI-A to Sec. VI-C . The implementation is written in C++ and run on an I7 Intel processor. The algorithm is also compared with the

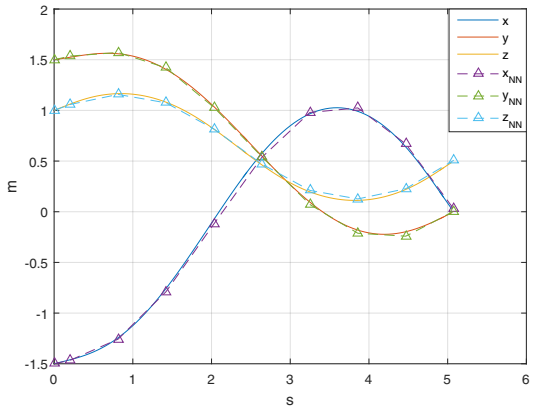
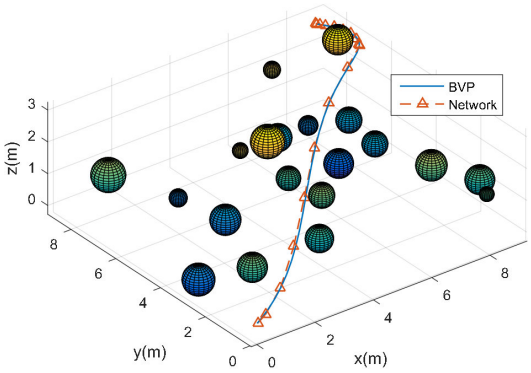
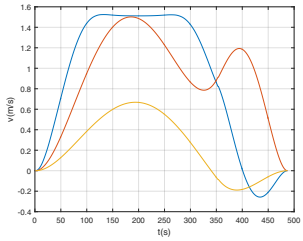


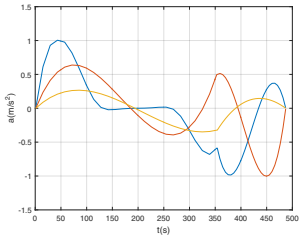
Fig. 1: Comparison between trajectory generated by the networks and the BVP solution



(a) Position



(b) Velocity



(c) Acceleration

Fig. 2: An example of kinodynamic BIT* with neural-network based local steering. (a) shows the comparison between neural network solution and a general bvp solver solution. (b) and (c) shows the recovered velocity and acceleration profile from network approximation with the state limit as $\mathbf{v}_{\max} = [1.5,\ 1.5,\ 1]$ and $\mathbf{a}_{\max} = [1,\ 1,\ 0.5]$.

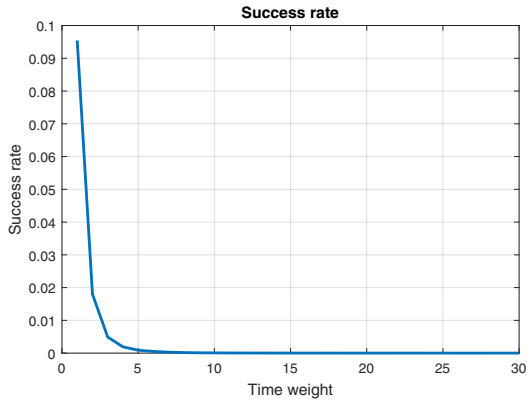


Fig. 3: Feasibility test for analytic local steering method.

one using analytic local steering method, which is further elaborated in Sec. VI-B.

A. Simulation

Fig. 2 shows an example of the simulation result. 20 obstacle-balls with variable size are randomly generated from a bonding box of $10m \times 10m \times 3m$. The start position is set as $[0.2, 0.2, 0.2]$ and end position is set as $[9, 9, 2.5]$. The initial velocity and acceleration, the end velocity and acceleration are all set to zero.

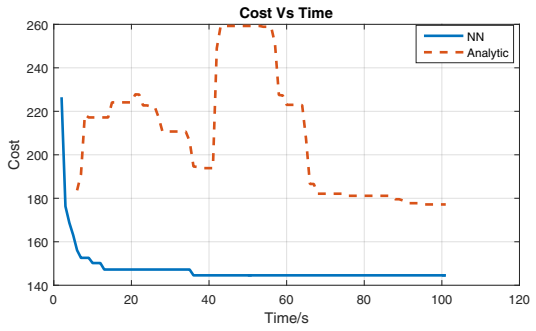


Fig. 4: The plot shows the average convergence rate of the kinodynamic BIT* planning over 10 runs.

B. Performance comparison with analytic local steering

There are some research work using analytic local steering for kinodynamic planning. [11] provides a general framework for kinodynamic planning for robots with controllable linear dynamics using RRT*. By fixing the initial state, end state and total time duration of the trajectory, closed-form solutions for optimal trajectories can be easily derived from Pontryagin minimum principles. By further taking the derivative of the cost function with respect to time and equal it to zero, the optimal time duration can be calculated accordingly. Note that the analytic solution of the controller actually no longer exists with the presence of state constraints. To address the problem, any violation of control inputs or state limits, would be viewed as a special collision of the motion edge of the tree and further discarded. The main idea of the analytic approach is to temporarily ignore the state constraints and focus on generating fast and computationally efficient motion primitives which can be easily tested on constraints violation later on.

However, the major problem is that most of generated motion primitives would turn out to be in-feasible if the state limit is relatively small, which is actually fairly common for indoor navigation application. As a result, there are only very limited amount of feasible motion primitives for a sampling-based planner to use, which causes the low converge rate. The situation gets worse when the time weight of performance index is higher, as show in the Fig. 3. For each time weight, 10,000 pairs of states are randomly sampled from a bounding box. The position is from $[-10, -10, 0]$ to $[10, 10, 3]$ and maximum velocity and acceleration are limited as $\mathbf{v}_{\max} = [1.5,\ 1.5,\ 1]$ and $\mathbf{a}_{\max} = [1,\ 1,\ 0.5]$ respectively. The generated motion primitives are tested against velocity and acceleration limits (we do not enforce the position constraints here), and the success/feasible rate then is further recorded. As shown, the success/feasible rate decrease significantly decreases while time weight increases.

The comparison of the convergence rate of the two methods are shown in Fig. 4. We use the same set of obstacles shown in Fig. 2 and conduct 10 runs for each steering method. For each run, the allowable planning time is 100 seconds, and the cost versus time profile is recorded accordingly. The averages of the cost are computed over 10

runs at the given times, hence there are jumps in the cost profile in Fig. 4. Clearly, the network-based steering finds initial solution much quicker than the one using analytic steering method. Also, the network-based one has faster convergence rate as well.

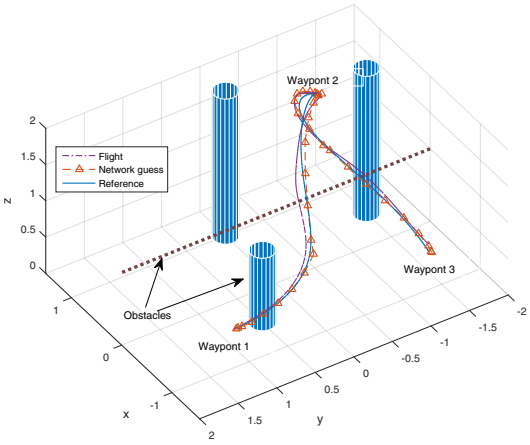
C. Flight test

The real flight experiment has also been conducted. The quadcopter uses the VICON motion capture system for the state estimation. The environment is pre-known and static and the allowable planning time is set as 10 seconds. The vehicle is approximated by a $40\,cm \times 40\,cm \times 40\,cm$ bounding box. The flight performance is shown in Fig. 5. The online video for the flight test can be found via the link `http://uav.ece.nus.edu.sg/kinoNN.html`.

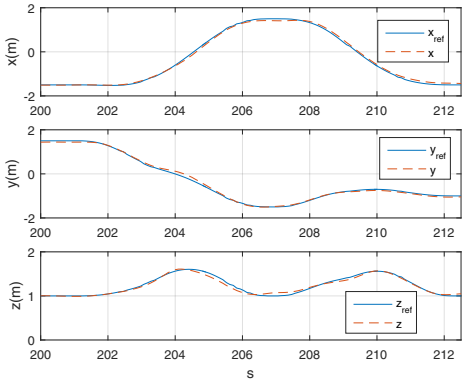
VII. Conclusion

This paper discusses the challenges of motion planning for quadcopters and presents a sampling-based kinodynamic planning algorithm with an MLP neural-network based local steering method. The main search planner is selected as BIT* for its fast convergence rate. A dynamic model of quadcopter is simplified into a tripple integrator. Then the discretized formulation and corresponding solution for local steering problem are proposed and discussed. An MLP network is further used to approximate the position profile, cost, and the time step duration of the steering trajectory. The potential of the proposed approach towards a real-time onboard implementation is illustrated by various simulations and a real flight test. The video for the flight test is available online, with the link as `http://uav.ece.nus.edu.sg/kinoNN.html`

(a) Experiment setup



(b) Trajectory



(c) Control performance

Fig. 5: Real flight test. (a) shows the experiment setup. (b) and (c) show the flight performance.

References

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

[2] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in Proceedings of the *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, pp. 1917–1922, Oct 2012.

[3] M. Hehn and R. D'Andrea, "Real-Time Trajectory Generation for Quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, Aug 2015.

[4] S. Lai, K. Wang, H. Qin, J. Q. Cui and B. M. Chen, "A robust online path planning approach in cluttered environments for micro rotorcraft drones," *Journal of Control Theory and Technology*, vol. 14, no. 1, pp. 83–96, Feb 2016.

[5] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in Proceedings of the *2016 IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, pp. 1476–1483, May 2016.

[6] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics*, vol. 4 (2), pp. 100107, 1968.

[7] S. K. Phang, S. Lai, F. Wang, M. Lan and B. M. Chen, "Systems design and implementation with jerk-optimized trajectory generation for UAV calligraphy," *Mechatronics*, vol. 30, pp. 65–75, Jun 2015.

[8] Gaël Guennebaud and Benoît Jacob and others, "Eigen v3", `http://eigen.tuxfamily.org`, 2010.

[9] S. Karaman, E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, pp. 846–894, 2011.

[10] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," Proceedings of the *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, pp. 3067–3074, May 2015.

[11] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," Proceedings of the *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, pp. 5054–5061, May 2013.

[12] C. Xie, J. van den Berg, S. Patil and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," Proceedings of the *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 2015, pp. 4187–4194, May 2015.

[13] R. Allen, M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," *2016 AIAA Conference on Guidance, Navigation and Control*, 2016.