2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

Planning Swift Maneuvers of Quadcopter Using Motion Primitives Explored by Reinforcement Learning

Efe Camci and Erdal Kayacan

Abstract— **In this work, we propose a novel, learning-based approach for swift maneuver planning of unmanned aerial vehicles using motion primitives. Our approach is composed of two main stages: learning a set of motion primitives during offline training first, and utilization of them for online planning of fast maneuvers thereafter. We propose a compact disposition of motion primitives which consists of roll, pitch, and yaw motions to build up a simple yet effective representation for learning. Thanks to this compact representation, our method retains an easily transferable, reproducible, and referable knowledge which caters for real-time swift maneuver planning. We compare our approach with the current state-of-the-art methods for planning and control, and show improved navigation time performance up to 25% in challenging obstacle courses. We validate our approach through software-in-the-loop Gazebo simulations and real flight tests with Diatone FPV250 Quadcopter equipped with PX4 FMU.**

I. Introduction

In this work, we develop a learning-based planner which enhances the navigation time performance of the current state-of-the-art methods by incorporating swift maneuvers. Unlike the other available planning approaches which focus on planning better (shorter, smoother, or higher-clearance) local or global paths [1]–[3], our concern is to explore desirable control sequences to plan swift motions which will ultimately decrease the overall navigation time.

The standard pipeline of motion planning is comprised of three stages such as finding a collision-free global path, optimizing/smoothing this path, and finding the required controls to follow the resultant path. The novel contribution of our approach presents itself in the last step with an intelligent control authority switch mechanism to generate fast maneuvers. We train an agent offline in virtual environment to learn an extensive set of maneuvers together with their corresponding control sequences using reinforcement learning (RL). The agent is then deployed in previously unseen environments to perform fast navigation by incorporating this retrospective knowledge. Once encountered with a particular maneuver during the operation, if the agent has previously learned how to perform it, the control authority switches to the agent to execute the maneuver. Otherwise, control authority stays on the default controller of the quadcopter to guarantee stability. We compare our method with a current state-of-the-art technique, i.e., geometric tracking control [4] over minimum snap trajectories [5] in challenging obstacle courses. We test our method both in Gazebo simulations and real flights, discussing computational complexity, control accuracy, and navigation time performance aspects together.

The rest of this work is organized as follows: Section II presents a literature review on agile motion planning of unmanned aerial vehicles (UAVs). Section III explains the modelling and identification processes rendering the acquired parameters of our quadcopter. Section IV introduces our novel approach in detail including motion primitives, learning process, and switch mechanism. Sections V and VI discuss the simulation and real flight test results, while Section VII draws the conclusion and states the future work.

II. Related Work

There are a variety of approaches for agile maneuver planning of UAVs. One of them is developed in [6] via sequential trajectory generation for aggressive flights. The method incorporates the mode switching concept to go beyond near-hover conditions. A similar concept is exploited in [7] for motion planning of combat aircrafts. Likewise in [8], complex aerobatic flights are planned using discrete maneuver sequences and safe transitions in between.

Apart from mode switching, mixed-integer programming is proposed for path planning of UAVs in [1], while this proposal is validated in [9] for aggressive flights with the aid of the model-based control approaches. On the other hand, a Lyapunov-based, linear time invariant controller is designed to enable high performance during aggressive flight in [10] while linear parameter-varying controller is introduced for high speed trajectory tracking in [11].

In parallel, the learning-based strategies yield promising results for swift maneuver planning of UAVs. The apprenticeship learning is employed in [12] and [13] to perform acrobatic maneuvers with a helicopter platform, including flip-flaps, tic-tocs, and stall turns. Both approaches mimic the demonstrations by an expert pilot. Furthermore in [14], the parameters of $a\ priori$ formulated flip maneuver are learned. In addition, iterative learning control is applied on quadcopters in [15] to improve horizontal, agile, point-to-point motions, while the approach is extended for a larger class of motions in [16].

Inspired by some of the aforementioned approaches, we develop a learning-based swift motion planning method in which there is no need for:

- simplification in the mathematical model,
- demonstration by expert pilots,
- $a\ priori$ formulation of the maneuvers.

Efe Camci is with School of Mechanical and Aerospace Engineering (MAE), Nanyang Technological University (NTU), 50 Nanyang Avenue, 639798, Singapore. Email: efe001@e.ntu.edu.sg

Erdal Kayacan is with Department of Engineering, Aarhus University, DK-8000 Aarhus C, Denmark. Email: erdal@eng.au.dk

978-1-5386-7926-5/$31.00 ©2019 AACC

III. Modelling and Identification

In this work, our platform of interest is Diatone FPV250 Quadcopter. It is described in Fig. 1 with the body and inertial coordinate frames, and rotor orientations. It has 6-DoF which stand for the 3 translational and 3 rotational motions. The forces acting on the vehicle are the thrust force created by the rotors in z_B direction and the gravitational force in -z_I direction. Consequently, Newton's equation of motion is stated as in [5]:

$$m\ddot{r} = -mg\mathbf{z}_I + \mathrm{F}_T\mathbf{z}_B \tag{1}$$

where g is the gravitational acceleration and r is the position vector for the quadcopter in the inertial frame, while F_T being the overall thrust provided by the rotors which is given as:

$$\mathrm{F}_T = \sum_{i=1}^{4} \mathrm{F}_i \tag{2}$$

where F_i stands for the thrust created by each rotor.

The moments acting on the vehicle, on the other hand, are the moments created around the x_B and y_B axes due to the distance of the motors to these axes and the moment produced around the z_B axis due to the anti-torque of each rotor. Consequently, Euler's equation for the rotational motion is stated as in [5]:

$$\dot{\omega} = I^{-1}\Big(-\omega \times I\omega + \begin{bmatrix} \mathrm{M}_\phi \\ \mathrm{M}_\theta \\ \mathrm{M}_\psi \end{bmatrix} \Big) \tag{3}$$

where I is the inertia matrix and ω is the angular velocity vector. The symbols M_ϕ, M_θ, M_ψ are the overall moments about each axis, and they are defined as:

$$\begin{bmatrix} \mathrm{M}_\phi \\ \mathrm{M}_\theta \\ \mathrm{M}_\psi \end{bmatrix} = \begin{bmatrix} (-\mathrm{F}_1 + \mathrm{F}_2 + \mathrm{F}_3 - \mathrm{F}_4)l \\ (-\mathrm{F}_1 + \mathrm{F}_2 - \mathrm{F}_3 + \mathrm{F}_4)l \\ (-\mathrm{M}_1 - \mathrm{M}_2 + \mathrm{M}_3 + \mathrm{M}_4) \end{bmatrix} \tag{4}$$

where l is the distance from the motors to the x_B and y_B axes, while M_i are the moments created due to the spinning of each rotor.

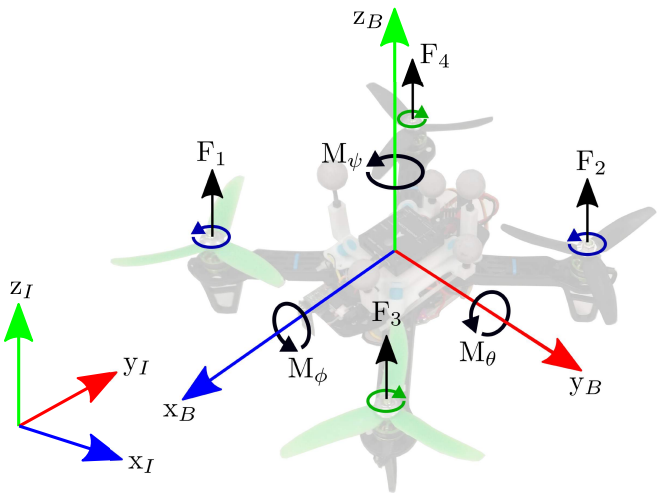

Fig. 1: Diatone FPV250 Quadcopter.

TABLE I: Quadcopter specifications. [17]

Symbol	Description	Value
m	Mass (kg)	0.414
l	Moment arm (m)	0.0884
I_{xx} I_{yy} I_{zz}	Moments of inertia (kgm^2)	0.0015 0.0015 0.0028
k_t	Thrust coefficient (Ns^2)	7.47×10^{-7}
k_m	Moment coefficient (Nms^2)	4.48×10^{-9}

The forces and moments produced by the rotors are related to the speeds of the rotors as follows:

$$\begin{aligned} \mathrm{F}_i &= k_t\Omega_i^2, \\ \mathrm{M}_i &= k_m\Omega_i^2 \end{aligned} \tag{5}$$

where Ω_i are the rotor speeds, while the parameters k_t and k_m are the thrust and moment coefficients, respectively. In order to identify these parameters, we conduct a sequence of experimental measurements, and match them with the simulation data. Besides, we create a thorough SolidWorks design of our platform based on [17] in order to obtain the mass and inertia properties precisely. Consequently, the parameters of our platform are obtained as in Table I.

IV. Motion Planning

The proposed motion planning approach is based upon offline learning of motion primitives which cater for designing swift maneuvers, and online utilization of them during the operation governed by a control authority switch mechanism. The approach benefits from the compact disposition of the proposed motion primitives both for offline learning and online planning, as it is explained in the following subsections.

A. Motion Primitives

The motion primitives proposed in this work are in the form of carefully selected 3 inputs, namely, roll, pitch, and yaw rate. Motivated by the nature of the control inputs given by an expert pilot for flying quadcopters with a transmitter, these 3 inputs are proposed to be in the shape of Bézier curve-based profiles. Bézier curves, firstly studied in [18], are parametric curves which have been widely used in computer graphics. Their mathematical basis is built upon Bernstein polynomials [19]. The mathematical representation of Bézier curves is given as:

$$\begin{aligned} &C(t) = \sum_{i=0}^{n} \binom{n}{i} P_i B_{n,i}(t), \quad B_{n,i}(t) = (1-t)^{(n-i)}\, t^i, \\ &t \in S_t, \quad S_t = \{x : x \in \mathbb{R},\ \ 0 \le x \le 1\} \end{aligned} \tag{6}$$

where $B_{n,i}(t)$ are the Bernstein polynomials of n^{th} degree and P_i are the control points, while the set S_t defines the region of interest for the parameter t. With the suitable choice of n and P_i, any curve can be represented in a parameterized manner exploiting Bézier curves.

We exploit Bézier curves to design input profiles for the elements of the motion primitives. For each distinguished maneuver, a Bézier curve profile is learned considering a certain criteria. We detail this process in the next subsection.

B. Learning

A combination of desirable values for the proposed motion primitive elements can move the quadcopter from one configuration to another. It might be compelling though, to find the combination of the optimal values for these inputs considering complex dynamics of quadcopter. Instead, we propose an RL-based framework to explore sub-optimal values for these inputs. RL not only discards the consideration of the 12-state equations of motion but also creates a compact knowledge which is easily transferable for future flights.

RL is a type of machine learning method which enables $artificial\ intelligence\ in\ robotics$ paradigm, broadly. It is based on the notion of learning-by-experience in which an agent gets a reward or punishment for its actions during particular situations by observing the changes in its environment as a consequence of its actions. The agent tries to assess its previous actions on the previous states in this way and make its future action selections more wisely. The essential aim of the agent is to explore the optimal actions which maximize the sum of the rewards [20].

In the batched simulations, we adapt incremental RL technique for the episodic motion primitive learning. Incremental RL is suitable for the episodic learning tasks [21], [22] since it provides a computationally efficient learning. The rewards obtained at each episode can be used directly for updating the Q values, without requiring an excessive memory [20]. Different from the conventional Q-learning, the Q-update rule for incremental RL is as follows [20]:

$$Q_{n+1}(s,a) = Q_n(s,a) + \frac{1}{n}\left(R_n - Q_n(s,a)\right) \tag{7}$$

where $Q_n(s,a)$ is the current Q value for the action a taken at the state s. The symbol n is the number of visits to the action a while R_n is the observed reward, and $Q_{n+1}(s,a)$ is the updated Q value for the state-action (s,a) pair.

In our proposed algorithm, an episode comprises the whole maneuver, rather than each discrete motion at each sampling time as in the conventional Q-learning. State s defines the maneuver characteristics and the state of the quadcopter right before the maneuver. In order to account for the aggressiveness of the maneuver, the state of the quadcopter is considered as its forward velocity u, right before the maneuver starts. Regarding the maneuver characteristics, it is defined by the turning angle, curvature, and altitude change during the turn. It is approximated via the fourth order Bézier curves to represent the maneuver in a parameterized manner by exploiting the control points as described in (6). State s is mathematically formulated as follows:

$$s \in S_u \times \mathbb{R}^{15}, \quad S_u = \{x : x \in \mathbb{R},\ \ u_{min} \leq x \leq u_{max}\} \tag{8}$$

where u_{min} and u_{max} are selected as 0.2 and 2.0m/s, respectively, considering the motion capabilities of our quadcopter.

Action a, on the other hand, stands for the attitude plans based upon the Bézier curve-based motion primitives. The action cluster is bounded to be in the reasonable region without the loss of the extensity of our approach. Motion primitives consisting of roll (ϕ), pitch (θ), and yaw rate ($\dot{\psi}$) profiles are sampled randomly within these bounds by exploiting the control points of Bézier curves. While these bounds can be chosen with a variety of different settings changing from fairly conservative to highly extreme; in this work, we define the interval of interest as follows:

$$\begin{aligned} &\phi \in S_\phi, \quad \theta \in S_\theta, \quad \dot{\psi} \in S_{\dot{\psi}}, \\ &S_\phi = \{x : x \in \mathbb{R},\ \ 0 \leq |x| \leq \phi_{max}\}, \\ &S_\theta = \{x : x \in \mathbb{R},\ \ 0 \leq |x| \leq \theta_{max}\}, \\ &S_{\dot{\psi}} = \{x : x \in \mathbb{R},\ \ 0 \leq |x| \leq \dot{\psi}_{max}\} \end{aligned} \tag{9}$$

where ϕ_{max} and θ_{max} are $90°$, while $\dot{\psi}_{max}$ is $360°$/s. By using these settings, an extensive motion set including the instantaneous turn around and full bank motions of the quadcopter is taken into consideration.

Reward R, lastly, represents the reward function for the agent to evaluate its actions. Our reward function is constructed upon three characteristics, i.e., maneuver execution time T, average deviation from path σ, and final deviation from local goal point δ. The mathematical details of the reward function are as follows:

$$\begin{aligned} &R = f(R_T, R_\sigma, R_\delta), \quad R_T, R_\sigma, R_\delta \in S_R, \\ &S_R = \{x : x \in \mathbb{R},\ \ 0 < x \leq 10\ \vee\ x = -100\} \end{aligned} \tag{10}$$

where R_T, R_σ, and R_δ represent the rewards obtained upon the evaluation of the aforementioned three characteristics. The bounds on the reward are determined by ensuring that a decent award for favorable actions and a harsh punishment for unfavorable ones are given. By this way, the probable crashes in dense environments are avoided during online utilization. The details of the reward function are given as:

$$R_T = \begin{cases} 10, & \text{for } T < T_{min} \\ \dfrac{k_T}{T}, & \text{for } T_{min} \leq T \leq T_{max} \\ -100, & \text{for } T_{max} < T, \end{cases} \tag{11}$$

$$R_\sigma = \begin{cases} 10, & \text{for } \sigma < \sigma_{min} \\ \dfrac{k_\sigma}{\sigma}, & \text{for } \sigma_{min} \leq \sigma \leq \sigma_{max} \\ -100, & \text{for } \sigma_{max} < \sigma, \end{cases} \tag{12}$$

$$R_\delta = \begin{cases} 10, & \text{for } \delta < \delta_{min} \\ \dfrac{k_\delta}{\delta}, & \text{for } \delta_{min} \leq \delta \leq \delta_{max} \\ -100, & \text{for } \delta_{max} < \delta \end{cases} \tag{13}$$

where upper and lower bounds for each characteristics are selected by trial-and-error as in Table II. The parameters k_T, k_σ, and k_δ are determined in a way that they provide continuity between the maximum reward given within the bounds of each characteristics and the globally maximum reward.

TABLE II: Design parameters.

Parameter	min	max
T	0.05s	5.0s
σ	0.001m	0.1m
δ	0.001m	0.1m

C. Control Authority Switch Mechanism

Despite the extensive capabilities of RL, it is wasteful to learn almost every possible motion primitive, i.e., even the ones for creating attitude plans for milder maneuvers. To account for this issue, we introduce an intelligent control authority switch mechanism in our approach. Our planning algorithm switches between two basic modes: near-hover motions and swift maneuvers. For the former, it utilizes the direct attitude plans enforced by the default geometric tracking controller. For the latter, however, it generates the attitude plans based upon the motion primitives learned. The seamless transition between two modes relies on the performance of both the desirable motion primitives and the default geometric tracking controller. While the desirable primitives mostly generate motion plans which allow for stable completion of maneuvers in addition to swift execution of turns, the precise geometric tracking controller takes over the control smoothly due to its abilities to deal with abrupt inputs.

We introduce a function $\xi(s)$, called $confidence\ level$, as the criteria for switching between two modes. As the name suggests, it is a parameter which indicates if the agent has already gained knowledge, i.e., is confident, in generating the learning-based attitude plans at the state s or not. Particularly, the agent checks whether an action a^* with a positive Q value is explored for at least one of the ϵ-close states to the state s^*. While the role of the function within the overall architecture can be seen in Fig. 2, its mathematical definition is elaborated as follows:

$$\xi : \mathbb{R} \times \mathbb{R}^n \longrightarrow \{0,1\}, \quad u \in \mathbb{R}, \quad \lambda \in \mathbb{R}^n,$$
$$\xi(u,\lambda) = \begin{cases} 1, & \text{if } \exists (u,\lambda) \in S_{past} \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

where λ are the trajectory data points for the maneuver and the set S_{past} refers to the states for which at least one action with positive Q value is explored. The definition of ϵ-closure is as follows:

$$\Delta u = |u^* - u|, \tag{15}$$

$$\Delta\lambda = \frac{\sum_{i=0}^{n} \mathrm{d}\Big(\lambda^*\big(\frac{i}{n}\big), \lambda\big(\frac{i}{n}\big)\Big)}{n}, \tag{16}$$

where n is the number of data points and $\mathrm{d}(\lambda^*, \lambda)$ is the Euclidean distance between the data points of the actual and target curves.

$$\begin{aligned} &\text{IF} \quad \exists s(u,\lambda): \quad \Delta u \le \epsilon_u \ \wedge \ \Delta\lambda \le \epsilon_\lambda, \\ &\text{THEN} \quad s^*(u^*,\lambda^*) \text{ is } \epsilon\text{-close.} \end{aligned} \tag{17}$$

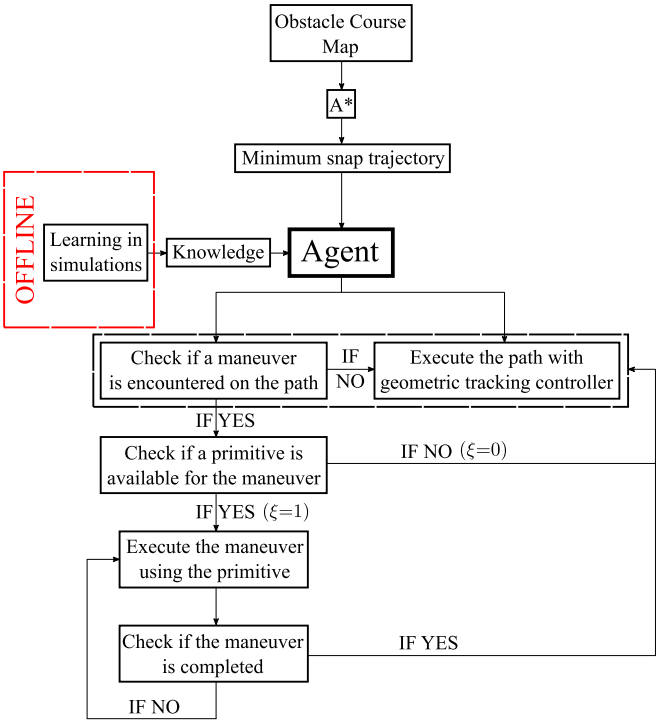

Fig. 2: Overall system architecture.

V. Simulations

In this work, in order to account for the safe learning, the $modus\ operandi$ consists of two main stages such as the learning in the batched simulations first, and the utilization of the knowledge thereafter. Since the quadcopter is an inherently unstable system, direct learning in dense environments without any initial, stable knowledge may lead to undesirable motions, even a crash. Therefore, we first create a realistic model of our quadcopter by exploiting the identified parameters in Section III, and conduct the initial learning process in an obstacle-free environment in Gazebo. In the sequel, we deploy our quadcopter to plan its path and corresponding motion by exploiting its retrospective knowledge.

A. Learning

During the learning stage, the agent spends maximum of 200 iterations for each distinguished maneuver in order to explore the best actions, i.e., the best-fit attitude plans for $\phi, \theta, \dot{\psi}$ regarding the reward function, by crafting the Bézier curve-based motion primitives. In particular, the agent tries to explore the favorable control points of the two types of Bézier curves as its actions, namely, linear and quadratic, governed by ε-greedy method.

As can be seen in Fig. 3, the agent creates attitude plans randomly within the specified bounds. The strengthened curves represent the primitives which are visited many more times since they are explored to yield high rewards. The other curves are explored to be undesirable for that particular maneuver once and punished with a negative reward.

Regarding the total training time, each maneuver is explored for 200 iterations in an epoch. An epoch lasts for approximately 400-500s on a standard Dell Precision M4800 laptop computer. The overall training time depends on maneuver types and diversity but roughly 60-70 maneuvers can be learned within approximately 8 hours with these settings.

B. Utilization of the Wisdom

Following the batched learning stage, we deploy our agent for the exploitation of the retrospective knowledge. We consider two case studies: a constant altitude operation and a 3D motion. We create complicated environments including challenging turns as can be seen in Figs. 4 and 5. For a comparison with the current state-of-the-art methods, we also include the results obtained using minimum snap trajectory generation [5] and finding the required controls by a geometric tracking controller [4]. In all cases, A*-search algorithm is used for finding a collision-free path, and this path is parameterized by the polynomials produced using quadratic programming to have a minimum snap trajectory.

In the first case study, the agent is deployed in a maze-like environment to navigate at a constant altitude. The nominal path is the same both for the agent and geometric tracking controller. The only difference is that the agent may plan fast motions using its retrospective knowledge if a previously learned state is encountered during the sharp turns. The results in the 3rd row of Fig. 4 show that the agent increases the overall navigation time performance by 25% via taking the turns up to 55% faster as depicted in the 1st row of Fig. 4. This improvement comes with a cost though, as seen in the 3rd row of Fig. 4, agile maneuvers may yield a little bit larger deviation. However, ultimately, they cater for decreasing the overall navigation time safely. It is also worth to mention that the agent plans swift maneuvers only for four turns despite the total number of turns being higher than four. When the agent encounters with a maneuver which is not explored during offline learning, it does not get the control authority thanks to the intelligent switch mechanism.

In the second case study, the agent is deployed in an obstacle course which resembles challenging drone-racing tracks. Besides sharp turns in the horizontal plane, this track requires sudden altitude changes which increase the difficulty level even more. When deployed, the agent is again able to decrease the overall navigation timeby utilizing the swift motion primitives. As depicted in the 1st row of Fig. 5, the agent generates agile motion plans which allow up to 62% of improvement in maneuver completion time. The aggregated enhancement in the navigation time performance eventually reaches up to 25%. In this case, however, the deviation from the path is comparably larger in z_I due to the difficulty of the maneuvers resulting from the rapid altitude changes. Nevertheless, the agent completes the task in the obstacle course safely. An important remark herein is that the nominal path is already parameterized to work the default geometric tracking controller on its limits. The vehicle reaches maximum speed of 6m/s in this configuration. In order to increase the navigation time performance of the geometric tracking controller, if we parameterize the path by decreasing the time required for completing *keyframes* even more, the vehicle fails to complete the track due to excessively large deviations which result in a crash.

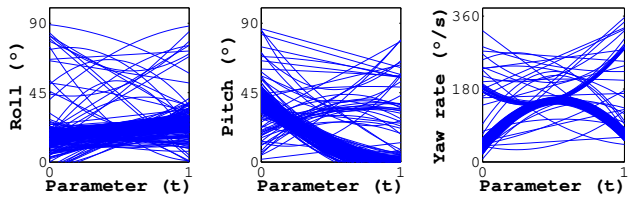

Fig. 3: Bézier curve-based motion primitives learned over 200 episodes.

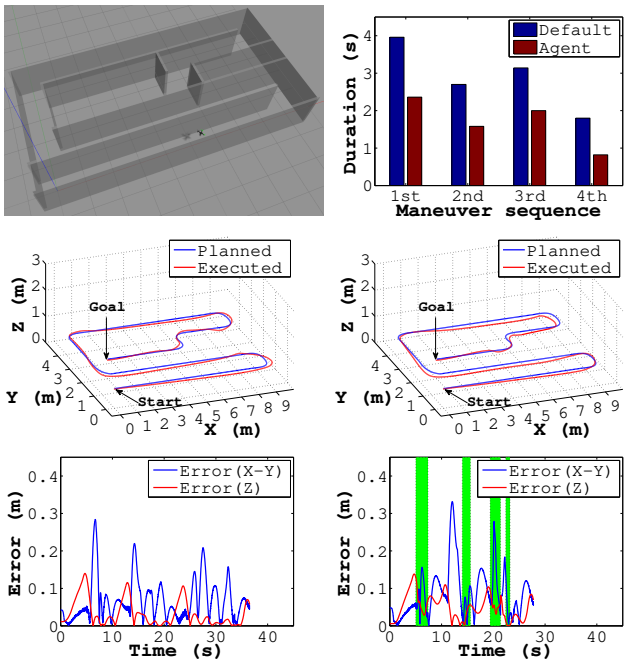

Fig. 4: **1st row:** Left: Maze-like environment. Right: Maneuver execution time. **2nd row:** Left: Geometric tracking controller's response. Right: Agent's response. **3rd row:** Left: Trajectory error for geometric tracking controller. Right: Trajectory error for the agent (The agent has the control authority within green-shaded areas.).

VI. Real Flights

In order to verify the *modus operandi* comprised of learning in virtual environment and utilization in real world, we test our approach on the Diatone FPV250 Quadcopter equipped with PX4 FMU in OptiTrack Lab at Nanyang Technological University. We create a challenging drone-racing kind of track as in the previous section including compelling turns and sudden altitude changes, but within 9m^2 area this time, due to the space limitations in our lab. We deploy the quadcopter to plan its motion by incorporating the retrospective knowledge gained in the simulation phase.

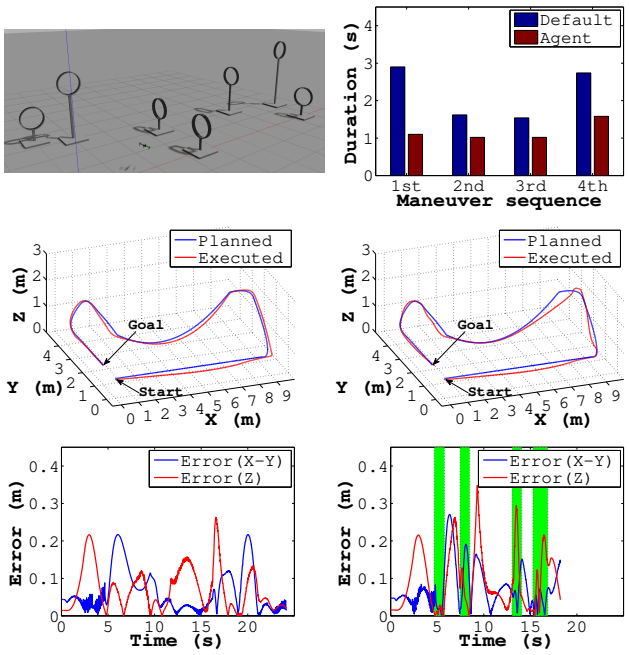

Fig. 5: **1st row:** Left: Challenging drone-racing kind of track. Right: Maneuver execution time. **2nd row:** Left: Geometric tracking controller's response. Right: Agent's response. **3rd row:** Left: Trajectory error for geometric tracking controller. Right: Trajectory error for the agent (The agent has the control authority within green-shaded areas.).

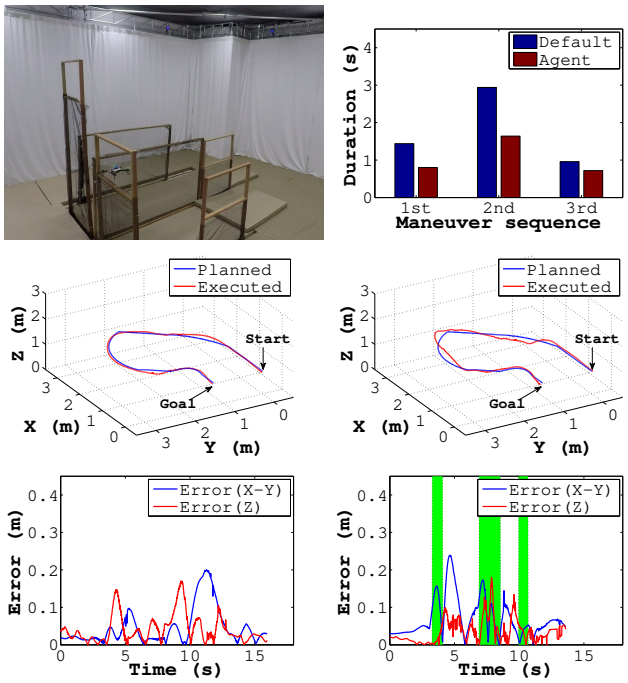

Fig. 6: **1st row:** Left: Real drone-racing kind of track. Right: Maneuver execution time. **2nd row:** Left: Geometric tracking controller's response. Right: Agent's response. **3rd row:** Left: Trajectory error for geometric tracking controller. Right: Trajectory error for the agent (The agent has the control authority within green-shaded areas.).

As can be seen in the 3rd row of Fig. 6, the agent decreases the overall navigation time by 15% in the real flight. The agent is able to take the turns up to 45% times faster than the default method as can be seen in the 1st row of Fig. 6, but when the aggregated navigation time is considered, the improvement is relatively less than the previous case studies. This is because the number of maneuvers in which the agent could plan swift motions is less in this case due to the limited lab space. As regards to the error, its range is comparably lower than the previous drone-racing kind of track in virtual environment because the limited area of our lab imposes lower velocities of quadcopter. The maximum velocity measured in real flights is 3m/s, which is the half of the value in the simulations. Therefore, the real vehicle yields more precise navigation as compared to the simulations. The video for the experiments can be found at: `https://youtu.be/PY4GYmtcG54`

As regards to the computational complexity, the number of motion primitives stored in the library is the most significant factor. The results averaged over 10 experiments with exponentially increasing number of primitives in the library are given in Fig. 7. The aforementioned standard laptop is used without any code optimization on C++ executables for these 10 experiments. As can be seen, 1,000,000 motion primitives can easily be searched, extracted, and used within the library without causing a delay in the operation with 50 Hz frequency. Therefore, the agent demonstrates sufficiency to be employed for real-time applications.

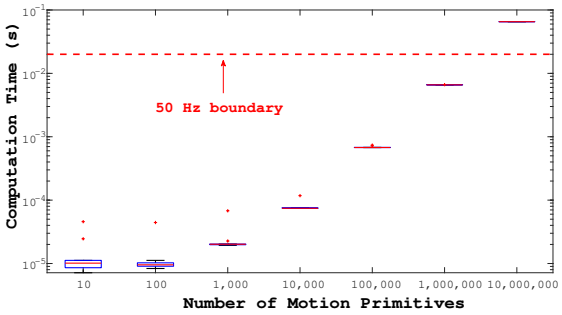

Fig. 7: Average computation time for different number of motion primitives in the library.

VII. Conclusion

In this work, we propose a novel approach for swift maneuver planning of quadcopters to increase navigation time performance in dense environments. We demonstrate the applicability of the proposed *modus operandi* comprised of learning in batched simulations first, and utilization in the environments of interest thereafter. The proposed approach shows a decent competency in increasing the navigation time performance by the motion primitive-based swift motion planning as compared to a current state-of-the-art method.

Looking forward, we are planning to conduct further learning in real-time flights. In this way, probable errors resulting from the system modelling and identification can be circum-

vented, and the best-fit motion plans can be explored for real vehicles. Moreover, we are also interested in investigating more advanced control authority switching techniques. In addition, we are also interested in developing more efficient learning strategies which can allow for generalization.

ACKNOWLEDGMENT

This work is financially supported by the Singapore Ministry of Education (RG185/17).

REFERENCES

[1] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 42–49.

[2] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[3] Y. Dong, E. Camci, and E. Kayacan, "Faster rrt-based nonholonomic path planning in 2d building environments using skeleton-constrained path biasing," *Journal of Intelligent & Robotic Systems*, pp. 1–15, 2017.

[4] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5420–5425.

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

[6] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[7] N. K. Ure and G. Inalhan, "Autonomous control of unmanned combat air vehicles: Design of a multimodal control and flight planning framework for agile maneuvering," *IEEE Control Systems*, vol. 32, no. 5, pp. 74–95, 2012.

[8] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1649–1654.

[9] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1469–1475.

[10] Y. Chen and N. O. Pérez-Arancibia, "Lyapunov-based controller synthesis and stability analysis for the execution of high-speed multi-flip quadrotor maneuvers," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 3599–3606.

[11] P. G. Cisneros, C. Hoffmann, M. Bartels, and H. Werner, "Linear parameter-varying controller design for a nonlinear quad-rotor helicopter model for high speed trajectory tracking," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 486–491.

[12] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[13] J. Tang, A. Singh, N. Goehausen, and P. Abbeel, "Parameterized maneuver learning for autonomous helicopter flight," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1142–1148.

[14] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1642–1648.

[15] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 1731–1736.

[16] A. P. Schoellig, F. L. Mueller, and R. DAndrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, pp. 103–127, 2012.

[17] L. Degiovanni and P. Bernard. (2016) Tp drone diatone 250 fpv. [Online]. Available: http://eduscol.education.fr/sti/ressources_pedagogiques/tp-drone-diatone-250-fpv#fichiers-liens

[18] P. De Casteljau, "Outillages méthodes calcul," *Andr e Citro en Automobiles SA, Paris*, 1959.

[19] G. G. Lorentz, *Bernstein polynomials*. American Mathematical Soc., 2012.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[21] E. Camci and E. Kayacan, "Waitress quadcopter explores how to serve drinks by reinforcement learning," in *Region 10 Conference (TENCON), 2016 IEEE*. IEEE, 2016, pp. 28–32.

[22] M. Mehndiratta, E. Camci, and E. Kayacan, "Automated tuning of nonlinear model predictive controller by reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3016–3021.