Learning motion primitives for planning swift maneuvers of quadrotor

Camci, Efe; Kayacan, Erdal

2019

Camci, E., & Kayacan, E. (2019). Learning motion primitives for planning swift maneuvers of quadrotor. *Autonomous Robots*, 43, 1733-1745. doi:10.1007/s10514-019-09831-w

<https://hdl.handle.net/10356/143523>

<https://doi.org/10.1007/s10514-019-09831-w>

© 2019 Springer Science+Business Media. This is a post-peer-review, pre-copyedit version of an article published in *Autonomous Robots*. The final authenticated version is available online at: <http://dx.doi.org/10.1007/s10514-019-09831-w>

Downloaded on 26 Apr 2023 22:54:32 SGT

Learning Motion Primitives for Planning Swift Maneuvers of Quadcopter Using Reinforcement Learning

Efe Camci · Erdal Kayacan

Received: date / Accepted: date

Abstract This work proposes a novel, learning-based method to leverage navigation time performance of unmanned aerial vehicles in dense environments by planning swift maneuvers using motion primitives. In the proposed planning framework, desirable motion primitives are explored by reinforcement learning. Two-stage training composed of learning in simulations and real flights is conducted to build up a swift motion primitive library. The library is then referred in real-time and the primitives are utilized by an intelligent control authority switch mechanism when swift maneuvers are needed for particular portions of a trajectory. Since the library is constructed upon realistic Gazebo simulations and real flights together, probable modeling uncertainties which can degrade planning performance are minimal. Moreover, since the library is in the form of motion primitives, it is computationally inexpensive to be retained and used for planning as compared to solving optimal motion planning problem algebraically. Overall, the proposed method allows for exceptional, swift maneuvers and enhances navigation time performance in dense environments up to 20% as being demonstrated by real flights with Diatone FPV250 Quadcopter equipped with PX4 FMU.

Keywords reinforcement learning · path planning · quadrotor · agile maneuvers

1 Introduction

In the past few decades, advances in unmanned aerial vehicle (UAV) research have grown remarkably. From design and development of innovative vehicles (Oosedo et al (2017); Fuller et al (2017)) to autonomous navigation and control aspects (Santamaria-Navarro et al (2017); Bareiss et al (2017); Mehndiratta and Kayacan (2017); Sarabakha et al (2018)), there is a rich literature history for UAV research. Despite this great development, the number of commercial applications of UAVs is still limited. In real world, UAVs are expected to operate in dense, populated, or dynamic environments. For safe and effective autonomy, the tasks to be accomplished are perception (Ling et al (2017)), state estimation (Mebarki et al (2017)), map construction (Korrapati and Mezouar (2017)), path planning (Dong et al (2018)), and control (Neunert et al (2016)).

The focus in this work is towards the planning side within the whole autonomy concept, planning swift maneuvers of UAVs in particular. Rapid task completion of UAVs is vital in real world applications such as inspection, surveillance, search and rescue in mines, within remainders of wrecked buildings after natural disasters, or in nuclear power plants after large scale accidents. In addition, rapid motion of UAVs is desired for recently emerging leisure activities such as drone racing. In all of these applications, UAVs may need to react to the sudden changes in their dynamic surroundings and have challenging motions within dense environments. Therefore, it is a necessity for UAVs to be capable of performing swift maneuvers. Nevertheless, planning this type of maneuvers might be compelling for the conventional motion planning methods. Since they mostly require the consideration of the full state equations of motion of UAVs for planning, the planning process is computationally demanding. Previous research addresses this issue by enabling simplifications in the equations of motion

Efe Camci
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798
E-mail: efe001@e.ntu.edu.sg

Erdal Kayacan
Department of Engineering, Aarhus University, DK-8000 Aarhus C
E-mail: erdal@eng.au.dk

(Hehn and D'Andrea (2011); Müller et al (2011)) for planning. However, the assumptions made for enabling simplifications, i.e., near-hover flight conditions, are not valid for agile maneuvers of UAVs.

Another line of research approaches to the problem from the aspect of finding a computationally inexpensive motion primitive (Mueller et al (2015)). Promising results are presented taking this approach by demonstrating the ability of evaluating around a million motion primitives per second on a standard laptop computer. On the other hand, the paper by Hehn and DAndrea (2015) proposes the use of a decoupled approach to reformulate the path planning problem considering separate trajectories in each translational degree of freedom. The researchers are able to plan trajectories of quadcopters from a large class of initial states to the target state at rest. The two approaches are yet to be integrated into a complete path planning framework as steering functions.

In the paper by Loianno et al (2017), the researchers exploit differential flatness of quadcopters (Mellinger and Kumar (2011)) and solve the aggressive maneuver planning problem using quadratic programming. In the paper by Deits and Tedrake (2015), the researchers again benefit from differential flatness but they use iterative regional inflation by semidefinite programming for higher level planning of aggressive flights. They validate the approach by experimental results with an increasing obstacle density level in the paper by Landry et al (2016).

A different approach is developed by Mellinger et al (2012) via sequential trajectory generation for aggressive maneuvers of quadcopters. Each sequential segments are related to the particular controllers for the distinctive flight modes and the transitions in between. Multi-modal flight planning is also exploited in the paper by Ure and Inalhan (2012) for the agile maneuvers of a combat aircraft. Instead of searching the whole state space for solving the planning problem, the search is focused into distinctive flight modes. Similarly in the paper by Gillula et al (2010), planning of complex aerobatic flights are decomposed into sequences of maneuvers with safe transitions in between.

The learning-based methods yield up-and-coming results for planning aggressive maneuvers of UAVs. In the papers by Abbeel et al (2010); Tang et al (2010), apprenticeship learning is employed for a wide-range of exceptional maneuvers with a helicopter, including stall turns, tic-tocs, and in-place flips. Both approaches try to benefit from the expert human pilot demonstrations and extract the characteristics of the aforementioned maneuvers. In the paper by Hwangbo et al (2017), reinforcement learning (RL) is used to extract policies which cater for a quadcopter to stabilize from harsh initializations, i.e., upside-down throwing, by swift recovery maneuvers. Furthermore, in the papers by Lupashin et al (2010); Lupashin and DAndrea (2012), the parameters of *a priori* formulated flip maneuver are

learned. Based on a five-step flip primitive, five free parameters which describe the maneuver are optimized. Nevertheless, all the aforementioned learning-based methods omit the obstacles in the environment.

In this work, a novel, learning-based approach is developed for swift maneuver planning of UAVs. In particular, a large set of motion primitives are proposed to be learned in an automated fashion to build up an extensive motion library which is to be referred when swift maneuvers are needed. Unlike the previous motion primitive-based (Mueller et al (2015)) or learning-based (Lupashin et al (2010); Lupashin and DAndrea (2012); Abbeel et al (2010)) approaches, the proposed method is not only developed for attractive, high-performance flights in an obstacle-free environment but also integrated into overall path planning by a novel control authority switch mechanism in order to decrease total navigation time while satisfying reasonable precision in dense environments. Moreover, as being different from the previous learning-based approaches (Pfeiffer et al (2016); Camci and Kayacan (2016)), learning is conducted not only in simulations but also in real flights in order to upgrade the quality of the motion library by mitigating probable modeling uncertainties which can degrade planning performance. The two key strengths of the proposed approach are:

- Instead of finding the corresponding state variables and required control inputs for the whole desired spatial trajectory by using differential flatness of quadcopter, the solution is searched (for the particular segments of the trajectory) within previously learned motion primitives. The swift maneuvers and decreased navigation time are demonstrated taking this approach.
- Instead of solving the optimal control problem for planning at each sampling time (which can be computationally demanding and even imprecise due to modeling uncertainties), the solution is searched in a more focused manner within the motion library developed by the retrospective knowledge without any computational burden.

The organization of this work is as follows: Section 2 states the specific problem addressed in this work. Section 3 explains the details of the quadcopter dynamics. Section 4 proposes the overall path planning framework. Sections 5 and 6 present the simulation and experimental results, respectively. In the end, Section 7 draws some conclusions on the whole work.

2 Problem Statement

Kinodynamic motion planning problem is defined as to find the required control inputs of a robot in order to drive it from a start configuration to a goal configuration while satisfying its dynamical constraints and avoiding obstacles in the environment (LaValle and Kuffner (2001)). In this context, the

specific problem considered herein is to find - to learn - the desired values of the carefully selected control inputs which enable quadcopter to fly from an initial configuration to a goal configuration in minimum time by obeying intermediate configuration constraints within collision-free space. In addition, since real-time applications of quadcopters are of interest in this work, it is a requirement to develop an approach which is computationally inexpensive and suitable for usual operating frequencies (50-100 Hz) of these vehicles (Meier et al (2012)).

3 Quadcopter Model

The quadcopter used herein is the Diatone FPV250 Quadcopter (see Fig. 1) which has 6 degrees of freedom (6-DoF) composed of kinematics and dynamics in 3 axes. The kinematic equations represent the translational freedom in three inertial axes, x_I, y_I, z_I . The dynamic equations denote the rotational freedom along three body axes, x_B, y_B, z_B .

3.1 Kinematics

Two individual coordinate systems, namely *body* and *inertial* frames, are used to describe the kinematics of the quadcopter in 3D. The body frame is the coordinate frame whose origin is located at the centre of gravity of the quadcopter while its x axis (x_B) is towards the front of the vehicle and z axis (z_B) is pointing in the direction of the thrust vector as in Fig. 1. Consequently, the y axis (y_B) follows the orientation defined by the right hand rule. The inertial frame is the coordinate frame whose origin is located on the ground while its z axis (z_I) is pointing towards the opposite direction of the center of Earth. Using the aforementioned two coordinate frames, the location of the quadcopter can be represented as x, y, z with respect to the inertial frame. The orthogonal transformation matrix R_{IB} between the two coordinate frames is given by (c:cos, s:sin):

$$R_{IB} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (1)$$

where Euler angles roll(ϕ), pitch(θ), and yaw(ψ) are defined as the tilting angles of the body frame with respect to the inertial frame in x, y, z directions, respectively.

3.2 Dynamics

There are two main forces acting on the vehicle: the gravitational force in $-z_I$ direction and the thrust force in z_B direction. Abiding by the rigid body assumption, Newton's

Fig. 1 Quadcopter schematic diagram.

equation of motion can be written as (Mellinger and Kumar (2011)):

$$m\ddot{r} = -mgz_I + F_T z_B \quad (2)$$

where r is the position vector of the center of gravity of the quadcopter denoted in the inertial frame and g is the gravitational acceleration. The symbol F_T is the total thrust created by propellers and it is given as:

$$F_T = \sum_{i=1}^4 F_i \quad (3)$$

where F_i describes the thrust created by each propeller.

Besides the aforementioned two forces, there are three main moments acting on the quadcopter as described in Fig. 1. The moments along x_B and y_B axes are due to the moment arms of the propellers from the center of the UAV while the moment along z_B axis results from the anti-torque effect of each propeller. Euler's equation of motion herewith can be written as (Mellinger and Kumar (2011)):

$$\dot{\omega} = I^{-1} \left(-\omega \times I \omega + \begin{bmatrix} M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} \right) \quad (4)$$

where ω is the angular velocity vector which consists of three components and I is the inertia matrix. The symbols M_ϕ, M_θ, M_ψ represent the total moments around each axis and F_T is the total thrust force. They are related to the propeller speeds Ω_i via the input allocation matrix which is given as:

$$\begin{bmatrix} F_T \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ -k_t l & k_t l & k_t l & -k_t l \\ -k_t l & k_t l & -k_t l & k_t l \\ -k_m & -k_m & k_m & k_m \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \quad (5)$$

Table 1 Quadcopter specifications. (Degiovanni and Bernard (2016))

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

where k_t and k_m are the thrust and moment coefficients while l is the vertical distance of the propellers to the x_B and y_B axes.

In the equations of both rotational and translational accelerations, the aerodynamic and gyroscopic terms are neglected because they require a substantial effort to be identified precisely by a sequence of experiments and simulations. The other parameters of the quadcopter are determined by creating a detailed SolidWorks design of the platform based upon the design by Degiovanni and Bernard (2016), and matching the real-time experiments with the data obtained in simulations. They are given as in Table 1.

4 Motion Planning

The overall motion planning in this work consists of two folds: offline learning of motion primitives within obstacle-free environments and online utilization of them for planning swift maneuvers in dense obstacle courses. The aim during offline learning is to build up an easily retainable, referable, and reproducible motion library which can be rapidly utilized in real-time.

4.1 Motion Primitives

The effectiveness and simplicity of the motion library strongly depend on the disposition of the motion primitives. In this work, this disposition is proposed to consist of carefully selected three inputs: roll, pitch, and yaw rate. Inspired by the nature of the transmitter inputs given by expert pilots during manual flights, these three inputs are proposed to have Bézier curve-based profiles. During a maneuver, the control inputs given by the pilot via transmitter resemble high order curve shapes. These shapes can be represented by best-fit curve techniques and mapped to particular states. As a result, an intelligent agent which can mimic the expert pilot can be acquired (Abbeel et al (2010)). In this work, these curves are explored not via the prior demonstration but with a random start to the exploration thanks to the effective representation by Bézier curves.

Bézier curves are parametric curves which have been used for modeling and evaluating smooth curves, especially in computer graphics. The mathematical basis of them (De Casteljaeu (1959)) is constructed upon Bernstein polynomials. A Bézier curve is represented via its control points and the related Bernstein polynomial expansion. The curve lies within the convex space formed by its control points. The degree of the Bernstein polynomials determines the smoothness of the curve. The mathematical formulation of a Bézier curve is stated as:

$$C : [0, 1] \rightarrow \mathbb{R}, \quad C(t) = \sum_{i=0}^n \binom{n}{i} P_i B_{n,i}(t), \quad (6)$$

where P_i are control points and $B_{n,i}(t)$ are Bernstein polynomials of n^{th} degree which are given as:

$$B_{n,i}(t) = (1-t)^{n-i} t^i. \quad (7)$$

Bézier curves can approximate any curve in a parameterized manner with the appropriate choice of n and P_i . They are exploited to craft input profiles for each element of the proposed motion primitives. During offline learning, a Bézier curve profile is explored for each distinguished maneuver while satisfying a certain criteria which is explained in the next subsection.

4.2 Reinforcement Learning

The motion primitives in the form of desirable Bézier curve profiles can move the quadcopter from one configuration to another. Nevertheless, it might be arduous to calculate the desirable combinations considering complex dynamics of quadcopter. Aside from the difficulty of the algebraic calculations resulting from 12-state equations of motion, the calculated solutions may be far from being applicable for the real vehicle if a modeling error exists. Instead, RL is proposed to be employed for exploring the desirable curve profiles of motion primitive elements. RL offers a favorable sub-optimality (if not the optimality) which caters for swift maneuver planning without the need of the complex algebraic calculations. In addition, RL can also minimize the performance degradations due to probable modeling errors since the learning is based on the realistic flights in Gazebo simulations together with the flights in real world.

RL is a nature-inspired, experience-based, simple learning method which enables the paradigm of *artificial intelligence in robotics*, broadly. In principle, it has three main elements such as *state*, *action*, and *reward*. Agent makes a selection among possible actions. It observes state and reward as a consequence of its action. It makes a new decision using them. The aim of the agent is to find the optimal actions which maximize the sum of the rewards (Sutton and Barto (1998)). Each of these three elements are pro-

posed proprietarily in this work to cater for relatively swift and reasonably precise maneuvers.

The state s is defined as a combination of two main characteristics such as the approaching velocity of the UAV right before the maneuver starts and the spatial trajectory of the maneuver. These two characteristics distinguish different maneuvers as different states. While we include the former directly as a number within the state representation, we incorporate the latter by using the control points (γ) of its 4th degree Bézier curve approximation. Consequently, the agent tries to find the unique optimal actions for a variety of different states. In order to create an extensive library, a number of different approaching velocities are considered together with a variety of spatial trajectories which are approximated by Bézier curves as described in (6). Overall, the state s is denoted as:

$$f: S_u \times \mathbb{R}^{15} \rightarrow \mathbb{R}, \quad f(u, \gamma) = s, \quad u \in S_u, \quad \gamma \in \mathbb{R}^{15}, \quad (8)$$

$$S_u = \{x: x \in \mathbb{R}, \quad u_{\min} \leq x \leq u_{\max}\}$$

where the parameters $u_{\min}$ and $u_{\max}$ are selected as 0.2 and 2.0m/s, respectively, by considering usual operating speeds of UAVs available in the lab.

The action a is defined as the low-level, attitude plans which are composed of roll (ϕ), pitch (θ), and yaw rate ($\dot{\psi}$). Each of them is in the form of Bézier curves. In particular, linear and quadratic Bézier curves are used to account for simplicity without the loss of generality. Again for simplicity in Q-learning, each action is represented by the control points of these curves which are randomly sampled within the search space of interest. The search space for the agent is bounded to avoid redundancy. In order to have a reasonably extensive motion set containing full bank and instantaneous turn around motions, the overall action cluster is determined as follows:

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

where the parameters $\phi_{\max}$ and $\theta_{\max}$ are selected as 1.57rad ($\approx 90^\circ$) while the parameter $\dot{\psi}_{\max}$ is chosen to be 6.28rad/s ($\approx 360^\circ/\text{s}$).

The reward R is the only guidance for the agent to assess its actions while trying to craft the desired input profiles by altering the control points of each primitive. It is defined as a combination of three criteria. These three criteria are proposed to be the maneuver execution time T , the average deviation from the path σ^1 , and the final deviation from the

local goal point δ . Individual rewards for each of these criteria form the overall reward function as follows:

$$g: S_R^3 \rightarrow \mathbb{R}, \quad g(R_T, R_\sigma, R_\delta) = R, \quad R_T, R_\sigma, R_\delta \in S_R \quad (10)$$

$$S_R = \{x: x \in \mathbb{R}, \quad 0 < x \leq 10 \vee x = -100\}$$

where R_T , R_σ , and R_δ represent the aforementioned, individual rewards. The rationale behind the bounds of each reward is to award the desirable actions in a decent manner while punishing the undesirable ones harshly. By this way, the agent is intended to be kept away from probable collisions in dense environments. The details of each element of the reward function are:

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

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

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

where the upper bounds denote the allowable values for each criteria for which a positive reward is given and the lower bounds define the values beyond which the rewards are saturated. In other words, as the value of a particular criteria gets lower than a saturation value, the reward increases dramatically while the performance is fairly the same. Therefore, the rewards given beyond a lower saturation bound are fixed as 10. The parameters k_T , k_σ , and k_δ , on the other hand, equate the maximum reward given within the reasonable region and the globally maximum reward. While all the aforementioned parameters can be chosen diversely based on the flexibility of the application, they are selected as in Table 2, in order to cater for swift maneuvers with a reasonable precision in dense environments.

Incremental RL is adapted for swift motion primitive exploration. It is a suitable choice because the RL problem addressed in this paper is formulated in a nonassociative setting in which agent learns how to behave for different states

Table 2 Design parameters for the reward function.

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

¹ Average deviation from the path is calculated by summing up the spatial distances between each data point of the 4th degree Bézier curve approximation of the desired path and the closest point on the 10th degree Bézier curve approximation of the resultant path to that point, and dividing them by the number of data points.

separately. It uses the evaluative feedback for active exploration as in common RL solution methods while avoiding the complexity of the full RL problem. In this setting, each complete maneuver corresponds to one episode. Each reward obtained at each episode is used directly in order to update Q values without requiring an excessive memory by the following formula (Sutton and Barto (1998)):

$$Q_{n+1}(s, a) = Q_n(s, a) + \frac{1}{n} (R_n - Q_n(s, a)) \quad (14)$$

where $Q_{n+1}(s, a)$ is the updated Q value for the state-action pair $s - a$ and $Q_n(s, a)$ is the current Q value before the update. The symbol n denotes the number of times that the agent has taken the action a at the state s while the resultant reward for the action is R_n .

During the learning process, actions are selected by ϵ -greedy method. The parameter ϵ herein describes the probability of selecting random actions while $1 - \epsilon$ denotes the probability of selecting the desirable actions learned through the episodes. The analysis for the suitable selection of ϵ -greedy policy is given in Section 5.

4.3 Switching Between Modes

In the proposed framework, the extensity of the learning process determines the number of situations for which swift maneuvers can be planned. Learning a larger set of motion plans for a broader set of states lets the agent to utilize the learning-based motion primitives on a wider range of situations, or vice versa. However, it is fairly demanding to learn all the possible motion primitives which cater for every single maneuver that the quadcopter can perform. Hence, it is a necessity to differentiate the maneuvers which have been learned and the ones which are unlearned. Therefore, a function ξ , called *confidence level*, is introduced. It is a binary function which indicates whether or not the agent is confident on creating learning-based motion plans for a particular maneuver. It governs a binary switching mechanism as described in Fig. 2. In detail, the function ξ indicates whether the ϵ -close states to the current state s^* are visited during the learning process and the Q value is positive for the action a^* taken during at least one of these ϵ -close states to the state s^* . The binary switching is provided accordingly, as in the following:

$$\begin{aligned} \xi : \mathbb{R} \times \mathbb{R}^n &\longrightarrow \{0, 1\}, \quad u \in \mathbb{R}, \quad \lambda \in \mathbb{R}^n, \\ \xi(u, \lambda) &= \begin{cases} 1, & \text{if } \exists(u, \lambda) \in S_{past} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (15)$$

where λ are the data points of the trajectory in the next keyframe and the set S_{past} contains the states which are ϵ -close to the states visited during learning and for which there

Fig. 2 Overall system architecture.

is at least one action explored with a positive Q value. The definition of ϵ -closure is as follows:

$$\Delta u = |u^* - u|, \quad (16)$$

$$\Delta \lambda = \frac{\sum_{i=0}^n d(\lambda^*(i), \lambda(i))}{n}, \quad (17)$$

where the $d(\lambda^*, \lambda)$ states the Euclidean distance between the data points of the curves while n represents the number of data points. Accordingly,

$$\begin{aligned} \text{IF } \exists(u, \lambda) : \Delta u \leq \epsilon_u \wedge \Delta \lambda \leq \epsilon_\lambda, \\ \text{THEN } s^* \text{ is } \epsilon\text{-close.} \end{aligned} \quad (18)$$

Overall, if the agent is confident on creating learning-based motion plans in a particular situation, it utilizes them for the maneuver. If it is unconfident, then the attitude plans enforced by the geometric tracking controller are utilized to account for safety. The overall system architecture of the proposed planning framework as applied to the quadcopter is shown in Fig. 2.

5 Simulation Studies

Initial batched simulations are conducted in Gazebo environment in order to build a reliable knowledge base for learning in real flights. A virtual representation of the quadcopter is created by exploiting the identified parameters of the Diatone FPV250 Quadcopter in Table 1. An extensive

set of maneuvers within an obstacle-free environment is performed/explored (depicted in Fig. 3). At each iteration, the agent selects a distinguished maneuver based upon its currently available knowledge, tries to perform it with the action decisions guided by ϵ -greedy method, and learns iteratively throughout the episodes.

For the learning in simulations, different ϵ -greedy policies are compared in terms of the quality of the knowledge together with the convergence pace. The comparison is based upon 10 epochs each of which is composed of 200 episodes as depicted in Fig. 4. In terms of the average reward value, the ϵ -greedy policy of 0.01 yields the most desirable results within the limited episode number of 200. The ϵ -greedy policy of 0.1 follows it with a little degraded average reward. The linear relaxations of ϵ from 1.0 to 0.1 and to 0.01 yield similar results to each other; they both have an increasing trend of average reward throughout the episodes. However, since both the number of episodes is constrained to bound the learning duration and the relaxation policies give an emphasis on exploration in the early episodes, they seem to have less average reward values at the end of an epoch. On the other hand, they yield superior performance regarding the averaged number of the desirable motion primitives acquired within 10 epochs. The reason behind this result is that the agent with the relaxation policies is more flexible to try and explore a variety of actions at the beginning of the epochs. The constant ϵ -greedy policies restrict their exploration capabilities relatively and give more importance to exploitation. As can be seen in Fig. 5, they could only obtain average desirable motion primitives of a number below 10. This number goes beyond 30 in the case of the relaxation policies while the relaxation policy from 1 to 0.1 yielding the highest number. Overall, it is a trade-off between achieving finely polished motion primitives in less amount and obtaining stable but not completely crafted primitives in more amount. Considering the fact that the initial Q look-up table acquired in simulations is to be iterated and the primitives are to be refined in real flights thereafter, the policy which can provide not fully refined but a larger number of desirable primitives is chosen. As a result, ϵ -greedy relaxation policy of 1 to 0.1 is selected for batched simulations as it is able to provide a vast and stable-enough initial primitives.

After finding the most suitable ϵ -greedy policy, batched learning simulations are continued. During this process, the agent spends minimum of 50 episodes for a maneuver. If a desirable primitive is not acquired within 50 episodes, the agent visits the same maneuver in the forthcoming epochs. On the other hand, if at least one initial stable primitive is acquired in 50 episodes, the agent avoids the selection of the same maneuver in the future epochs in order to avoid redundancy. Overall, the agent converges to a particular set of motion plans which are stored in a Q look-up table by

Fig. 3 Diatone FPV250 Quadcopter learning swift maneuvers in Gazebo.

Fig. 4 Average reward over episodes with different ϵ -greedy policies.

Fig. 5 Averaged number of desirable motion primitives learned through 10 epochs with different ϵ -greedy policies.

mapping the desired actions to the corresponding maneuvers. As regards to the total training time, an epoch composed of 50 episodes lasts for approximately 100-120s on a standard Dell Precision M4800 laptop computer. The total training time is highly subject to the number and type of the maneuvers considered but if the simulation is kept running for approximately 8 hours, roughly 60-70 different maneuvers can be explored.

6 Experiments with Real Quadcopter

Real flight experiments with the Diatone FPV250 Quadcopter equipped with PX4 FMU are conducted in OptiTrack Lab at Nanyang Technological University. They consist of two stages such as learning in the obstacle-free environment and test in the obstacle course. In the former, the prelim-

Fig. 6 Overall architecture of the real-time setup.

inary Q look-up table acquired in simulations is taken as a baseline and improved throughout the repetitive maneuvers by the real quadcopter. This stage is conducted offline within the obstacle-free environment in order to avoid possible crashes which may occur while the agent is exploring undesirable primitives that can yield large deviations from the nominal path. In the latter, the quadcopter is deployed for the challenging scenarios including sharp turns, narrow corridors/passages, and sudden altitude changes.

The overall system architecture for the real-time setup is explained in Fig. 6. The whole communication is achieved over Robot Operating System (ROS). Two ROS nodes run simultaneously on the Dell Precision M4800 laptop computer. The position and orientation information of the quadcopter is provided by the OptiTrack motion capture system at 240 Hz. Together with this information, the commands generated by the learning-based planner are conveyed to PX4 FMU through WiFi at 50 Hz.

6.1 Learning

The learning in real flights leads up for the possible improvements on exploring the best-fit motion primitives for swift maneuver planning. Although the initial motion primitives gained in simulations are stable enough to be used for performing a variety of maneuvers in an obstacle-free environment, the vehicle occasionally exceeds the allowable deviation limits defined in Section 4.2 for a positive reward. These results stem from the fact that there are a number of hardware features which can hinder the perfect representation of the real vehicle in Gazebo simulations such as battery power, electronic speed controller (ESC) performance, motor and propeller efficiency, nonlinear rotor dynamics in extreme attitude configurations, or delay in communication with the vehicle, etc. Therefore, further exploration of the motion primitives on the real vehicle can allow for captur-

Fig. 7 Diatone FPV250 Quadcopter learning swift maneuvers within an obstacle-free environment.

ing the real vehicle dynamics and planning more desirable motions.

The learning stage in real flights (Fig. 7) is a more conservative version of the learning phase in simulations to account for safety. The search space is focused around the stable motion primitives by restricting the actions within 0.1rad ($\approx 6^\circ$) of random walks for roll and pitch, and within 0.1rad/s for yaw rate. The ϵ -greedy relaxation of 1 to 0.1 is kept as the same. During an epoch, the relaxation is continued for the first 10 episodes, while it is fixed as 0.1 for the next 10 episodes until the following battery change of the vehicle. The aim of the learning stage in real flights is to evaluate and validate the retrospective primitives for the real vehicle, and improve them if possible.

As can be seen in Fig. 8, learning in real flights results in an increased average reward at the end of 10 epochs. The agent captures finely tuned primitives with upgraded reward values by exploring within the aforementioned random walks generated around the desirable primitives learned in simulations. Although the agent yields negative rewards in the beginning due to a relatively higher emphasis on exploration governed by adaptive ϵ -greedy policy from 1 to 0.1, it extracts the useful information from the repetitive flights which ultimately cater for exploring the best-fit primitives with upgraded rewards.

Fig. 8 Average reward in real flights over 10 epochs with the relaxed ϵ -greedy policy from 1 to 0.1.

It is also worth to mention that even relatively conservative random walks (0.1 rad) around the desirable motion primitives might change the flight performance significantly, as can be inferred from negative rewards. This result indicates the sensitivity of the overall motion to the inputs. It also reveals that the possible, slight deviations in the motion plans generated by the conventional planning methods due to probable modeling mistakes might result in hazardous failures. Therefore, the proposed learning-based method presents itself as an attractive choice for exploring the real dynamics and planning swift maneuvers more confidently in the obstacle course.

6.2 Test in the Obstacle Course

Following the learning phases in simulations and real flights, the vehicle is deployed to perform a rapid navigation from start point to goal point in three different obstacle courses. The first one includes two turns with altitude changes and a narrow corridor in between, as can be seen in Fig. 9. The second one evaluates the repetitive performance of the agent for swift 180° turns for four laps as in Fig. 10, while the third one challenges the agent by four turns with sudden altitude changes which vary from 0.6m to 1.2m, and evaluates the repetitive performance within four laps, as depicted in Fig. 11. In all the three case studies, the performance is compared with a current state-of-the-art method in which the overall path is generated using minimum snap trajectory generation (Mellinger and Kumar (2011)) and the required controls are found by geometric tracking controller (Lee et al (2010)) which is known as a competent choice for agile maneuvers. In all cases, a collision-free path is found in 3D using A*-search algorithm before it is parameterized in each axis by the polynomials obtained using quadratic programming to have a minimum snap trajectory.

In the first scenario, the agent is able to decrease the total navigation time by around 15% via planning two swift maneuvers through the path. The nominal path in 3D is planned and parameterized as the same in terms of the time

Fig. 9 **Top row:** The obstacle course of the first scenario including two turns and a narrow corridor on the left, maneuver completion time comparison on the right. **Second row:** Overlay of the planned and executed paths for geometric tracking controller on the left, for the proposed agent on the right. **Third row:** Trajectory errors for geometric tracking controller on the left, for the proposed agent on the right (green-shaded areas for the time intervals when the agent has the control authority). **Bottom row:** Navigation time comparison.

required to complete *keyframes* as described in the paper by Mellinger and Kumar (2011) both for the agent and geometric tracking controller. Only difference between these two cases is that the agent has the flexibility of planning swift motions using its retrospective knowledge. In other words, when a previously learned state is encountered during sharp turns, control authority switches to the agent. Accordingly, the agent plans swift maneuvers and takes the turns faster. The agent takes the first turn around 55% faster while this ratio is 35% for the second turn. This agility comes with a cost though. The agent yields a higher deviation from the nominal path, especially in the z_I -axis. This is mainly due to the fact that while the explored elements of roll, pitch, and yaw rate are commanded to the vehicle, an additional thrust input has to be commanded too. It is regulated with a proportional-derivative controller in the current version of the algorithm whose performance is highly dependent on the power available in the battery of the vehicle. This problem occasionally occurs for real vehicles since it is difficult to maintain the same level of voltage for every flight. Therefore, rectifying this problem will constitute a part of the future work of the authors as explained in the next section.

Fig. 10 Top row: The obstacle course of the second scenario including two 180° turns to be navigated at a constant altitude for 4 laps on the left, maneuver completion time comparison on the right. **Second row:** Overlay of the planned and executed paths for geometric tracking controller on the left, for the proposed agent on the right. **Third row:** Trajectory errors for geometric tracking controller on the left, for the proposed agent on the right (green-shaded areas for the time intervals when the agent has the control authority). **Bottom row:** Navigation time comparison.

In the second scenario, the focus is on a particular type of turn (180°) and the repetitive performance of the agent. In this case, the agent is able to utilize its retrospective knowledge to plan its swift motions within four laps and decrease the total navigation time by around 20%. It takes the turns up to 45% quicker as compared to geometric tracking controller. The overall navigation time improvement is more in this case as compared to the former one because the agent has a larger number of opportunities to utilize its swift motion plans within four laps. As regards to the quantification of the trajectory error, if the swift maneuvers are not conducted, the overall path can be considered as a mild one which enables geometric tracking controller to yield precise trajectory results. The agent, on the other hand, results in a relatively erroneous but moderately accurate trajectory. The agent completes the navigation without a crash.

In the third scenario, the agent is deployed in an obstacle course which resembles compelling drone-racing tracks. The agent is again able to improve the overall navigation time performance by utilizing the motion primitives learned during offline training. The improvement in the total naviga-

Fig. 11 Top row: The obstacle course of the third scenario including four turns and sudden altitude changes to be navigated for 4 laps on the left, maneuver completion time comparison on the right. **Second row:** Overlay of the planned and executed paths for geometric tracking controller on the left, for the proposed agent on the right. **Third row:** Trajectory errors for geometric tracking controller on the left, for the proposed agent on the right (green-shaded areas for the time intervals when the agent has the control authority). **Bottom row:** Navigation time comparison.

tion time is around 15% within four laps by taking the turns up to 65% faster. The deviation from the path stays moderate for x_I and y_I axes, but the aforementioned problem is present for the z_I -axis. Overall, the proposed method demonstrates adequacy to stay fairly on the nominal x_I and y_I while performing swift maneuvers for decreased total navigation time. Another important remark herein is that although the environment includes sixteen turns in four laps, the agent plans swift maneuvers only for thirteen of them. This is due to the intelligent control authority switch mechanism. When the agent encounters with a maneuver which remains unexplored after the offline learning, control authority stays on the default geometric tracking controller. This is also a part of the reason for relatively less enhancement on the navigation time performance in this scenario as compared to the second scenario.

As regards to the computational complexity, since the agent only needs to retain the retrospective knowledge instead of conducting complex algebraic calculations as the other nonholonomic planners, the size of the previously obtained data is the most influential parameter on the over-

Table 3 Averaged computation time comparison between the default method and the agent.

Method	Computation time (s)
Default method	7.09×10^{-5}
Agent with 10 primitives	4.35×10^{-5}
Agent with 100 primitives	4.99×10^{-5}
Agent with 1,000 primitives	6.15×10^{-5}
Agent with 10,000 primitives	8.77×10^{-5}
Agent with 100,000 primitives	3.73×10^{-4}
Agent with 1,000,000 primitives	0.0022
Agent with 10,000,000 primitives	0.0199

Fig. 12 Averaged computation time for different number of motion primitives in the library.

all computation time of the proposed method. For investigation of computational complexity, ten experiments are conducted with exponentially growing number of randomly generated primitives. The whole computation is conducted in the aforementioned standard laptop. The C++ executables created for the corresponding ROS nodes are directly implemented without any code optimization. The results averaged over these ten experiments show that the proposed method requires less computation power than the default method up to the presence of 1,000 primitives in the library as can be seen in Table 3. The results also indicate that the proposed method shows real-time applicability easily up to the presence of 1,000,000 motion primitives in the library as can be seen in Fig. 12. The computation time reaches and occasionally exceeds the limit of 0.02s (50Hz of common operational frequency) through ten experiments when 10,000,000 primitives are present in the library. Therefore, these results demonstrate the computational feasibility of the proposed method for real-time applications.

7 Conclusion

This work presents a novel, learning-based framework for planning swift maneuvers of quadcopters. In contrast to the conventional planning methods in which the full state equa-

tions of motion are considered for each sequential states algebraically, the approach is based upon creating a motion library which consists of the desirable motion primitives acquired by the retrospective knowledge in simulations and real flights. These motion primitives are later used when the vehicle is deployed in an obstacle course if the similar maneuvers are needed to be planned and executed.

The approach shows competency in planning swift maneuvers without the loss of safety in dense environments. The overall navigation time from the start to the goal in the environment is decreased up to 20% by these swift maneuver plans. The experimental validation of the proposed two-stage learning is demonstrated. The agent is able to learn the preliminary motion primitives in Gazebo simulations without any prior knowledge first, and improve upon them by continuing to learn in real flights thereafter. With the aid of real flight learning, the agent is able to alleviate the probable errors which may result from the uncertainties in the hardware and extract the best-fit motion plans.

This work also discusses the different learning strategies in the aim of exploring the most suitable learning type for quadcopters which are inherently unstable flying machines and which may end up with a crash unless the learning is conducted with a rigorous attention. It is concluded that in the first stage of learning, more relaxed policies together with relaxed boundaries can be used since the aim is to explore a higher number of preliminary motion primitives and since there is no danger of crashing in simulations. In the second stage of learning, a more conservative learning is to be conducted without giving a chance of unstable maneuver trials which may end up with the loss of the vehicle.

There are a few limitations of the approach which will constitute a part of the authors' future work. Firstly, the coarse switching between the agent's motion plans and the controls enforced by geometric tracking controller will be upgraded. Currently, it is just a binary switch mechanism which directly gives the authority either to the agent or to the default controller. This might yield abrupt motions in some cases. Especially while recovering after a swift maneuver, directly switching back to the default controller might not be fast enough to stabilize the vehicle. In order to cater for smooth transitions between modes, a more advanced switching mechanism which incorporates an intermediate planning between modes will be investigated. Secondly, the effect of the battery power available on the overall motion of the quadcopter will be investigated since it restricts the authors to use only fully charged batteries currently and still yields degraded performance in the z_I -axis. As the commands sent to the quadcopter consist of thrust and attitude plans for the execution of the learning-based motion primitives, it is observed that the execution performance of the thrust command changes drastically for different battery power levels. The last drawback of the proposed approach is that the agent

has to explore the desirable motion primitives for each type of maneuver unless the maneuvers are ϵ -close as defined in Section 4.3. It would be beneficial if a further research can enable knowledge transfer which can cater for reasoning for the maneuvers which are not explicitly learned and are not ϵ -close to the previously learned maneuvers. The authors are particularly interested in this branch of the future work because the initial learning effort can be decreased substantially in this way.

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

References

- Abbeel P, Coates A, Ng AY (2010) Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29(13):1608–1639
- Bareiss D, Bourne JR, Leang KK (2017) On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles. *Autonomous Robots* pp 1–16
- Camci E, Kayacan E (2016) Waitress quadcopter explores how to serve drinks by reinforcement learning. In: *Region 10 Conference (TENCON)*, 2016 IEEE, IEEE, pp 28–32
- De Casteljau P (1959) *Outillages méthodes calcul*. Andr e Citro en Automobiles SA, Paris
- Degiovanni L, Bernard P (2016) Tp drone diatone 250 fpv. URL <http://eduscol.education.fr/>
- Deits R, Tedrake R (2015) Efficient mixed-integer planning for uavs in cluttered environments. In: *Robotics and Automation (ICRA)*, 2015 IEEE International Conference on, IEEE, pp 42–49
- Dong Y, Camci E, Kayacan E (2018) Faster rrt-based non-holonomic path planning in 2d building environments using skeleton-constrained path biasing. *Journal of Intelligent & Robotic Systems* 89(3-4):387–401
- Fuller SB, Teoh ZE, Chirarattananon P, Pérez-Arancibia NO, Greenberg J, Wood RJ (2017) Stabilizing air dampers for hovering aerial robotics: design, insect-scale flight tests, and scaling. *Autonomous Robots* pp 1–19
- Gillula JH, Huang H, Vitus MP, Tomlin CJ (2010) 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, pp 1649–1654
- Hehn M, D’Andrea R (2011) Quadcopter trajectory generation and control. *IFAC Proceedings Volumes* 44(1):1485–1491
- Hehn M, DAndrea R (2015) Real-time trajectory generation for quadcopters. *IEEE Transactions on Robotics* 31(4):877–892
- Hwangbo J, Sa I, Siegwart R, Hutter M (2017) Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*
- Korrapati H, Mezouar Y (2017) Multi-resolution map building and loop closure with omnidirectional images. *Autonomous Robots* 41(4):967–987
- Landry B, Deits R, Florence PR, Tedrake R (2016) Aggressive quadrotor flight through cluttered environments using mixed integer programming. In: *Robotics and Automation (ICRA)*, 2016 IEEE International Conference on, IEEE, pp 1469–1475
- LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20(5):378–400
- Lee T, Leoky M, McClamroch NH (2010) Geometric tracking control of a quadrotor uav on se (3). In: *Decision and Control (CDC)*, 2010 49th IEEE Conference on, IEEE, pp 5420–5425
- Ling Y, Kuse M, Shen S (2017) Edge alignment-based visual-inertial fusion for tracking of aggressive motions. *Autonomous Robots* pp 1–16
- Loianno G, Brunner C, McGrath G, Kumar V (2017) Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters* 2(2):404–411
- Lupashin S, DAndrea R (2012) Adaptive fast open-loop maneuvers for quadcopters. *Autonomous Robots* 33(1-2):89–102
- Lupashin S, Schöllig A, Sherback M, D’Andrea R (2010) A simple learning strategy for high-speed quadcopter multi-flips. In: *Robotics and Automation (ICRA)*, 2010 IEEE International Conference on, IEEE, pp 1642–1648
- Mebarki R, Lippiello V, Siciliano B (2017) Vision-based and imu-aided scale factor-free linear velocity estimator. *Autonomous Robots* 41(4):903–917
- Mehndiratta M, Kayacan E (2017) Receding horizon control of a 3 dof helicopter using online estimation of aerodynamic parameters. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* p 0954410017703414
- Meier L, Tanskanen P, Heng L, Lee GH, Fraundorfer F, Pollefeys M (2012) Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots* 33(1-2):21–39
- Mellinger D, Kumar V (2011) Minimum snap trajectory generation and control for quadrotors. In: *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on, IEEE, pp 2520–2525
- Mellinger D, Michael N, Kumar V (2012) Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research* 31(5):664–674

- Mueller MW, Hehn M, D'Andrea R (2015) A computationally efficient motion primitive for quadcopter trajectory generation. *IEEE Transactions on Robotics* 31(6):1294–1310
- Müller M, Lupashin S, D'Andrea R (2011) Quadcopter ball juggling. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, pp 5113–5120
- Neunert M, de Crousaz C, Furrer F, Kamel M, Farshidian F, Siegwart R, Buchli J (2016) Fast nonlinear model predictive control for unified trajectory optimization and tracking. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, pp 1398–1404
- Oosedo A, Abiko S, Konno A, Uchiyama M (2017) Optimal transition from hovering to level-flight of a quadrotor tail-sitter uav. *Autonomous Robots* 41(5):1143–1159
- Pfeiffer M, Schaeuble M, Nieto J, Siegwart R, Cadena C (2016) From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. *arXiv preprint arXiv:160907910*
- Santamaria-Navarro A, Loianno G, Solà J, Kumar V, Andrade-Cetto J (2017) Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors. *Autonomous Robots* pp 1–18
- Sarabakha A, Fu C, Kayacan E, Kumbasar T (2018) Type-2 fuzzy logic controllers made even simpler: From design to deployment for uavs. *IEEE Transactions on Industrial Electronics* 65(6):5069–5077
- Sutton RS, Barto AG (1998) *Reinforcement learning: An introduction*, vol 1. MIT press Cambridge
- Tang J, Singh A, Goehausen N, Abbeel P (2010) Parameterized maneuver learning for autonomous helicopter flight. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, pp 1142–1148
- Ure NK, Inalhan G (2012) Autonomous control of unmanned combat air vehicles: Design of a multimodal control and flight planning framework for agile maneuvering. *IEEE Control Systems* 32(5):74–95