Automated Tuning of Nonlinear Model Predictive Controller by Reinforcement Learning

Mohit Mehndiratta, Efe Camci, and Erdal Kayacan

***Abstract*— One of the major challenges of model predictive control (MPC) for robotic applications is the non-trivial weight tuning process while crafting the objective function. This process is often executed using the trial-and-error method by the user. Consequently, the optimality of the weights and the time required for the process become highly dependent on the skill set and experience of the user. In this study, we present a generic and user-independent framework which automates the tuning process by reinforcement learning. The proposed method shows competency in tuning a nonlinear MPC (NMPC) which is employed for trajectory tracking control of aerial robots. It explores the desirable weights within less than an hour in iterative Gazebo simulations running on a standard desktop computer. The real world experiments illustrate that the NMPC weights explored by the proposed method result in a satisfactory trajectory tracking performance.**

I. Introduction

Model predictive control (MPC) has recently gained popularity for the control of aerospace systems, including multicopter unmanned aerial vehicles (UAVs) [1]–[5]. MPC is able to simultaneously handle constraints and optimize performance in a systematic and elegant manner via repetitive online optimization [6]. Howbeit, one of the major obstacles related to its implementation is its non-trivial tuning process. Typically, MPC design procedure involves selection of weighting parameters which reflect relative importance of minimizing each variable in objective function. Mostly, they are adjusted using the heuristic trial-and-error methods. However, this cumbersome process takes significant amount of trial time, while the quality of the obtained weights is highly dependent on the skill set of the designer.

Many successful works have been reported in the literature to tune MPC weights in a systematic way. For instance, online/offline tuning rules, which are based on the frequency domain analysis of the closed-loop MPC behavior, are proposed in [7]. This method first identifies the most dominating tunable parameters and later analyzes their effect on the overall closed-loop performance. Another approach for MPC tuning is given in [8], where an easy-to-use tuning formulation is deduced. However, their approach is only restricted to linear MPC without constraints, which further requires the plant representation in a first order plus dead-time model. This type of model representation is not suitable for representing multicopter UAVs as their dynamics are of higher order as well as open-loop unstable. In addition to offline tuning strategies, one amongst a few online tuning methods is presented in [9], where an online adaptive tuning strategy is proposed for the constrained linear MPC. In this work, analytical expressions for computing the closed-loop response sensitivity to the MPC tuning parameters are devised. However, this method is not applicable to UAVs as their dynamics are highly nonlinear wherein the closed-loop response gradients cannot be represented in terms of analytical expressions.

Apart from model-based tuning methodologies, machine learning techniques have also been explored for MPC tuning. In [10], a heuristic online tuning method for nonlinear MPC (NMPC) is presented where the gradient computation to determine the time propagation of the MPC parameters is replaced with fuzzy logic. However, utilizing fuzzy logic requires in-depth understanding of the underlying system dynamics which may not be available for new or inexpert practitioners. Recently, two other articles [11], [12] have adopted reinforcement learning (RL) for NMPC tuning as applied to UAVs. In the former, RL is utilized to tune NMPC for various flight conditions that are encountered within load transportation tasks, while in the latter, it is exploited for computing NMPC weights wherein NMPC is used to perform trajectory planning along with collision avoidance. Nevertheless, both approaches are far from lending themselves to tune NMPC weights completely from scratch. In addition, the applicability of these approaches for real world experiments are yet to be investigated.

Unlike the tuning methods above, the main objective in this study is to devise a technique which can substantially reduce the time and effort to be dedicated by the human user for (N)MPC tuning for real robotic systems. Thus, as the main contribution, we propose a novel RL-based framework for exploring the desirable (N)MPC weights in an automated manner. Thanks to the adaptive disposition of RL, the proposed framework is platform-independent and can be utilized for tuning any robotic system ranging from ground to aerial robots. As a case study, we demonstrate the proposed method for tuning the weights of NMPC algorithm which is employed for high-level position tracking control of a quadrotor UAV. First, we create a realistic Gazebo model of the UAV, and then, conduct iterative learning for the exploration of the desirable NMPC weights on a standard desktop computer for less than an hour. Thereafter, we proceed with utilizing these weights on the real vehicle without any post-tuning, and demonstrate a satisfactory trajectory tracking performance.

Mohit Mehndiratta and Efe Camci are with School of Mechanical and Aerospace Engineering (MAE), Nanyang Technological University (NTU), 50 Nanyang Avenue, 639798, Singapore. Email: mohit005@e.ntu.edu.sg; efe001@e.ntu.edu.sg.

Erdal Kayacan is with the Department of Engineering, Aarhus University, Aarhus 8200, Denmark. Email: erdal@eng.au.dk.

This study is organized as follows: Section II discusses the quadrotor dynamic model. Section III illustrates the NMPC problem formulation along with the RL-based tuning framework. Section IV presents statistical results for learning over batched simulations and a qualitative benchmark study on them. Thereafter, Section V validates the applicability of the explored NMPC weights in real world experiments. Lastly, Section VI draws conclusions from this study.

II. System Modeling

The UAV used in this work is a quadrotor with 'x-configuration'. It is considered as a rigid-body having two rotors in the front – front right rotor rotating counter-clockwise and front left rotor rotating clockwise – and two rotor at the back – back right rotor rotating clockwise and back left rotor rotating counter-clockwise – as shown in Fig. 1.

The translational kinematic equations, describing position of the UAV, are obtained using the transformation from body frame ($\mathscr{F}_B$) to Earth-fixed frame ($\mathscr{F}_E$) [13]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{EB} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{1}$$

where x, y, z represent the translational position which is defined in frame $\mathscr{F}_E$; u, v, w are the translational velocities that are defined in frame $\mathscr{F}_B$; R_{EB} is the translation transformation matrix between frames $\mathscr{F}_E$ and $\mathscr{F}_B$, and is expressed as ($c:cos,\ s:sin,\ t:tan$):

$$R_{EB} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\psi c\phi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \tag{2}$$

where ϕ, θ, ψ represent rotational attitude of the UAV defined in frame $\mathscr{F}_E$.

On the other hand, the rigid-body dynamic equations are derived based on the Newton-Euler formulation in the body coordinate system. The quadrotor is assumed to be a point mass, wherein all the forces act at the CG [13]:

$$\dot{u} = rv - qw + g\sin(\theta) + \frac{1}{m}F_x, \tag{3a}$$

$$\dot{v} = pw - ru - g\sin(\phi)\cos(\theta) + \frac{1}{m}F_y, \tag{3b}$$

$$\dot{w} = qu - pv - g\cos(\phi)\cos(\theta) + \frac{1}{m}F_z, \tag{3c}$$

where F_x, F_y, F_z are the total external forces acting on the quadrotor body in frame $\mathscr{F}_B$. These external forces are generated when a rotor (or propeller) is propelled through the surrounding air. Using the momentum theory, a hovering rotor can be modeled as: $F_i = K_F\Omega_i^2$, where F_i is the force generated by i^{th} rotor with Ω_i angular velocity. In addition, K_F is some positive intrinsic parameter of the rotor and is commonly referred as the force coefficient. Moreover, with respect to the quadrotor configuration shown in Fig. 1, the expressions for total external force (F_{ext}) acting in $\mathscr{F}_B$ frame, is written as:

$$F_{ext} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}. \tag{4}$$

The constant intrinsic parameters for the DJI F330 FlameWheel UAV in our laboratory are listed in Table I. While the parameters such as mass (m) and arm length (l) are directly measured, the thrust coefficient is evaluated based on a simple experimentation. It involves measuring thrust generated for various rotor RPMs and finding a simple relation between thrust and RPM values using data fitting techniques. One may refer [14] for details.

For the real-time dynamic optimization framework, the nonlinear quadrotor model is discretized based on direct multiple shooting method, utilizing a shooting grid size of $\Delta t_s = 0.01s$. In addition, the explicit Runge-Kutta 4$^{\text{th}}$ order integrator, with 2 steps per shooting interval is incorporated. Finally, the discretized nonlinear model of the quadrotor UAV can be represented as:

$$\mathtt{x}_{k+1} = \mathtt{f_d}(\mathtt{x}_k, \mathtt{u}_k), \tag{5}$$

$$\mathtt{z}_k = \mathtt{h}(\mathtt{x}_k, \mathtt{u}_k), \tag{6}$$

where the state and control vectors, $\mathtt{x} \in \mathbb{R}^{n_\mathtt{x}}$ and $\mathtt{u} \in \mathbb{R}^{n_\mathtt{u}}$ are:

$$\mathtt{x} = [x, y, z, u, v, w]^T, \quad \mathtt{u} = [\phi, \theta, \psi, F_z]^T, \tag{7}$$

and $\mathtt{z} \in \mathbb{R}^{n_\mathtt{x}}$ is the measurement vector. Also, the state and measurement functions are denoted by $\mathtt{f_d}(\cdot,\cdot)$:$\mathbb{R}^{n_\mathtt{x}} \times \mathbb{R}^{n_\mathtt{u}} \longrightarrow \mathbb{R}^{n_\mathtt{x}}$ and $\mathtt{h}(\cdot,\cdot)$:$\mathbb{R}^{n_\mathtt{x}} \times \mathbb{R}^{n_\mathtt{u}} \longrightarrow \mathbb{R}^{n_\mathtt{x}}$, respectively.

Remark 1: Since the presented algorithm is not utilized to perform the low-level (attitude) control for the UAV, the rotational kinematic and dynamic equations are omitted in this study.

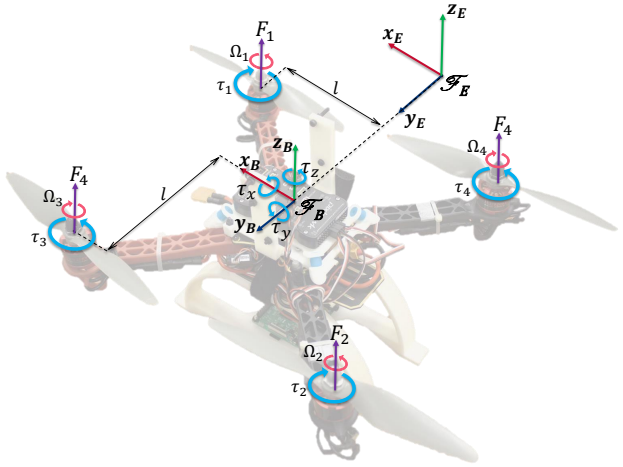

Fig. 1: Coordinate frame and sign conventions for the considered quadrotor UAV. Additional notation: τ_i is the torque generated by i^{th} rotor; τ_x, τ_y, τ_z are the external moments about x-, y- and z-axes respectively.

TABLE I: DJI F330 FlameWheel quadrotor intrinsic parameters.

Par.	Description	Value
m	Mass of quadrotor UAV	1.25kg
l	Moment arm	0.233m
K_f	Aerodynamic force coefficient	$4.209 \times 10^{-4}\text{N}-\text{s}^2$

III. Reinforcement Learning-Based Tuning of Nonlinear Model Predictive Controller

A. Nonlinear Model Predictive Control

(N)MPC computes an optimum current control action by optimizing model's behavior over a finite prediction horizon (N_c). The optimal forecast of the system behavior results from an open-loop online optimization, represented in the form of a constrained, finite horizon, optimal control problem (OCP). The solution of this open-loop OCP for the current system's state and at the current time instant results in an optimal sequence of control actions; the first term of which is then regarded as the optimal control action [6].

For our trajectory tracking application similar to [15], we formulate the parametric OCP for NMPC in the form of a least square function that penalizes the deviations of predicted state and control trajectories from their specified references, over the given prediction horizon window ($t_j \leq t \leq t_{j+N_c}$). In discrete time, the optimization problem is of the form:

$$\min_{\mathtt{x}_k,\mathtt{u}_k} \quad \frac{1}{2}\Bigg\{ \sum_{k=j}^{j+N_c-1} \left(\left\| \mathtt{x}_k - \mathtt{x}_k^{\text{ref}} \right\|_{W_\mathtt{x}}^2 + \left\| \mathtt{u}_k - \mathtt{u}_k^{\text{ref}} \right\|_{W_\mathtt{u}}^2 \right) + \left\| \mathtt{x}_{N_c} - \mathtt{x}_{N_c}^{\text{ref}} \right\|_{W_{N_c}}^2 \Bigg\} \tag{8a}$$

$$\text{s.t.} \quad \mathtt{x}_j = \hat{\mathtt{x}}_j, \tag{8b}$$

$$\mathtt{x}_{k+1} = \mathtt{f}_\mathtt{d}(\mathtt{x}_k, \mathtt{u}_k), \qquad k = j, \cdots, j+N_c-1, \tag{8c}$$

$$\mathtt{x}_{k,\min} \leq \mathtt{x}_k \leq \mathtt{x}_{k,\max}, \qquad k = j, \cdots, j+N_c, \tag{8d}$$

$$\mathtt{u}_{k,\min} \leq \mathtt{u}_k \leq \mathtt{u}_{k,\max}, \qquad k = j, \cdots, j+N_c-1, \tag{8e}$$

where $\mathtt{x}_k \in \mathbb{R}^{n_\mathtt{x}}$ is the differential state, $\mathtt{u}_k \in \mathbb{R}^{n_\mathtt{u}}$ is the control input and $\hat{\mathtt{x}}_j \in \mathbb{R}^{n_\mathtt{x}}$ is the current state estimate; time-varying state and control references are denoted by $\mathtt{x}_k^{\text{ref}}$ and $\mathtt{u}_k^{\text{ref}}$, respectively; terminal state reference is $\mathtt{x}_{N_c}^{\text{ref}}$; $W_\mathtt{x} \in \mathbb{R}^{n_\mathtt{x} \times n_\mathtt{x}}$, $W_\mathtt{u} \in \mathbb{R}^{n_\mathtt{u} \times n_\mathtt{u}}$ and $W_{N_c} \in \mathbb{R}^{n_\mathtt{x} \times n_\mathtt{x}}$ are the corresponding weight matrices, which are assumed constant for this application. Furthermore, $\mathtt{x}_{k,\min} \leq \mathtt{x}_{k,\max} \in \mathbb{R}^{n_\mathtt{x}}$ and $\mathtt{u}_{k,\min} \leq \mathtt{u}_{k,\max} \in \mathbb{R}^{n_\mathtt{u}}$, specify the lower and upper bounds on the states and control inputs, respectively.

Implementation of NMPC for Position Tracking: Utilizing the current feedback of other states, NMPC computes the optimized solution for the control input in terms of total thrust and attitude angles. This optimized solution is then passed to the low-level controller as a desired setpoint. Moreover, the low-level attitude controller is selected as a proportional-integral-derivative (PID) controller, which is designed individually for each axis with the control vector $\mathtt{u}_{\text{PID}} = [\Omega_1, \Omega_2, \Omega_3, \mu]^T$. Moreover, the overall control scheme is summarized within the UAV block of the schematic diagram shown in Fig. 2.

The state, control, and measurement vectors for the position tracking NMPC are composed of:

$$\mathtt{x}_{\text{NMPC}} = [x, y, z, u, v, w]^T, \quad \mathtt{u}_{\text{NMPC}} = [\phi, \theta, \psi, F_z]^T, \tag{9a}$$

$$\mathtt{z}_{\text{NMPC}} = \mathtt{x}_{\text{NMPC}} \tag{9b}$$

and the following state and control reference trajectories are selected for the optimization problem of NMPC:

$$\mathtt{x}^{\text{ref}} = [x_r, y_r, z_r, 0, 0, 0]^T, \quad \mathtt{u}^{\text{ref}} = [0, 0, 0, mg]^T, \tag{10}$$

where m and g are mass and gravitational constant, respectively. It is to be noted that for the OCP formulation of NMPC, the parameterization of the nonlinear model (in translation) is also done with respect to p, q and r. Hence, the three rotational rates are fed to NMPC along with the other states, as also illustrated in Fig. 2. Some input constraints are imposed in the definition of NMPC in order to achieve a stable behavior from the low-level controller:

$$0.5mg \text{ (N)} \leq F_z \leq 1.5mg \text{ (N)}, \tag{11a}$$

$$-35 \ (^\circ) \leq \phi \leq 35 \ (^\circ), \tag{11b}$$

$$-35 \ (^\circ) \leq \theta \leq 35 \ (^\circ). \tag{11c}$$

While the weight matrices $W_\mathtt{x}$ and $W_\mathtt{u}$ are tuned by RL, the terminal weight W_{N_c} is selected as: $W_{N_c} = 1.3 \times W_\mathtt{x}$, in order to maintain the stability of NMPC. Furthermore, the prediction window $N_c = 30$, is selected to facilitate the real-time applicability of the control framework.

B. Reinforcement Learning

RL is a nature-inspired, experience-based, learning method which 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, and it makes a new decision using them. The aim of the agent is to find the actions which maximize the rewards [16].

In this work, we adapt incremental RL technique for episodic learning of NMPC weights. Incremental RL is a desirable choice for episodic learning tasks [17] since it provides computationally efficient learning, as the rewards obtained in each episode can be used directly for updating the Q-values, without the need of an excessive memory [16]. The Q-update rule for incremental RL is given as:

$$Q_{n+1}(s,a) = Q_n(s,a) + \frac{1}{n}\left(R_n - Q_n(s,a)\right), \tag{12}$$

where $Q_n(s,a)$ is the current Q-value for the action a taken at the state s. The symbol n represents 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, state s, is defined as the flight mode of the quadrotor. There are two flight modes of interest such as 'hover' and 'move-to-a-setpoint' in this work. In hover mode, the vehicle tries to maintain its original position, whereas in move-to-a-setpoint mode, it tries to navigate to a pre-defined setpoint as fast as possible. Action a, on the other hand, is defined as the NMPC weight sets ($W_\mathtt{x}$, $W_\mathtt{u}$) where the action space is bounded as follows:

$$\begin{aligned} W_\mathtt{x} &\in \text{diag}([\mathscr{O}(10^2)]^{1\times 3}, [\mathscr{O}(10^{-1})]^{1\times 3}), \\ W_\mathtt{u} &\in \text{diag}([\mathscr{O}(10^2)]^{1\times 3}, \mathscr{O}(10^{-2})), \end{aligned} \tag{13}$$

while $\mathscr{O}$ represents the order of magnitude in the above equation. As regards to reward R, it is defined by certain

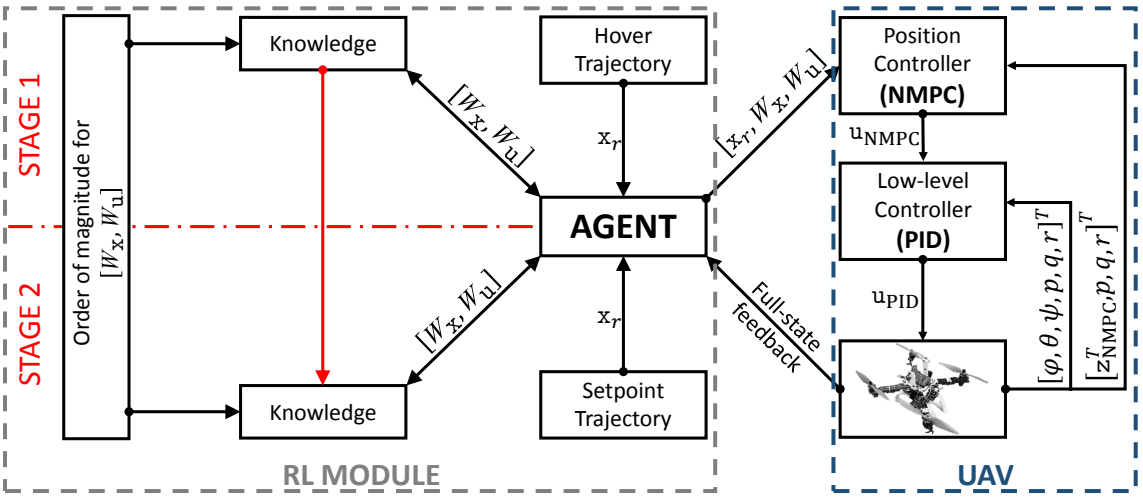

Fig. 2: RL-based tuning framework for NMPC which is employed as a position controller for the quadrotor UAV.

characteristics such as position error, rate of change of position error, jerk, or combinations thereof, details of which are provided in the Section III-B.

Implementation of the Learning Algorithm: Learning, which is represented by the RL module block within Fig. 2, mainly consists of two stages. In the first stage, the quadrotor is deployed with hover mode, and a broad set of weights which can hold the vehicle in the air safely are explored. In the second stage, the quadrotor is deployed with move-to-a-setpoint mode, and new weights are learned for a desirable transient response of the vehicle by directing the search towards the weights obtained in the first stage[1]. In this vein, while the weight sets explored in the first stage provide rough guidelines for the second stage, the second stage functions as the ultimate decision-maker to converge to a few set of weights which result in a favorable flight performance during trajectory tracking. The performance is evaluated based on the reward function which is expressed as:

$$\begin{aligned} &R = f(R_e, R_{\dot{e}}, R_{jerk}, R_{e_{ss}}), \\ &R_e, R_{\dot{e}}, R_{jerk}, R_{e_{ss}} \in S_R, \\ &S_R = \{x : x \in \mathbb{R},\ 1 \leq x \leq 100 \vee x = -100\}, \end{aligned} \tag{14}$$

where R_e, $R_{\dot{e}}$, R_{jerk}, and $R_{e_{ss}}$ represent the rewards obtained upon the evaluation of the four characteristics, i.e., position error, rate of change of position error, jerk, and steady-state position error, respectively. The details of each element of the reward function are given as:

$$R_E = \begin{cases} 100, & \text{for } E < \dfrac{E_{max}}{100} \\ \dfrac{E_{max}}{E}, & \text{for } \dfrac{E_{max}}{100} \leq E \leq E_{max} \\ -100, & \text{for } E_{max} < E, \end{cases} \tag{15}$$

where E indicates the criteria of interest. While the parameters E_{max} can be chosen diversely based on the specific needs of the applications, they are selected for two different stages as given in Tables II and III in this work. We set the first three parameters in an acceptable region for stable hover in Stage 1, and we relax them in Stage 2 for move-to-a-point configuration because those criteria generally have higher values when the vehicle navigates. The only parameter restricted when proceeding from Stage 1 to 2 is e_{ssmax} since it yields similar values for two different flight modes, and it is desired to search for better NMPC weights in Stage 2, based upon the ones obtained in Stage 1.

TABLE II: Design parameters for Stage 1.

Parameter (max)	x	y	z
Position error	0.1m	0.1m	0.05m
Velocity error	0.01m/s	0.01m/s	0.005m/s
Jerk	0.5m/s^3	0.5m/s^3	0.5m/s^3
Steady state position error	0.1m	0.1m	0.05m

TABLE III: Design parameters for Stage 2.

Parameter (max)	x	y	z
Position error	0.25m	0.25m	0.1m
Velocity error	0.2m/s	0.2m/s	0.12m/s
Jerk	5m/s^3	5m/s^3	8m/s^3
Steady state position error	0.05m	0.05m	0.03m

The maximum number of episodes for both stages is kept as the same. Actions are governed by ε-greedy method. While the ε-greedy policy is initialized as 1.0 and annealed linearly with the episode number for the first half of each stage, it is kept as constant for the second half [18]. The states and actions together with their respective rewards are stored in a Q look-up table whose maximum dimension is equal to total number episodes since the episodic learning is employed in this work.

IV. Simulations

For the iterative simulations, we create a realistic model of our quadrotor UAV in Gazebo by exploiting the identified parameters in Section II. We repeat each simulation ten times in order to statistically validate our approach. We compare four different reward functions which are proposed, examined, and assessed throughout the search for the best

[1]New weights are searched within 10% variance of the desirable weights obtained in Stage 1 (see the red arrow in Fig. 2).

evaluation criteria for NMPC weight tuning in this study. After obtaining the desirable weight sets, we benchmark them qualitatively for a square-shaped trajectory tracking.

A. Learning

Through the search for the best assessment criteria of NMPC weights, we generate four different reward function candidates. Starting from position error, we enrich the reward function by adding rate of change of position error, jerk, and steady state position error criteria at each step gradually. We assign these reward functions to four different agents:

$$R_i = \begin{cases} f(R_e), & \text{for } i = 1 \\ f(R_e, R_{\dot{e}}), & \text{for } i = 2 \\ f(R_e, R_{\dot{e}}, R_{jerk}), & \text{for } i = 3 \\ f(R_e, R_{\dot{e}}, R_{jerk}, R_{e_{ss}}), & \text{for } i = 4, \end{cases} \tag{16}$$

where R_i represents the reward function for the Agent$_{(i)}$.

The agents are deployed for two-stage learning. Each stage includes 50 episodes. The actions of the agents are governed by the ε-greedy policy which is initialized as 1.0 and annealed to 0.5 for Stage 1, 0.3 for Stage 2. The motivation for this selection is to converge to only a few sets of highly desirable NMPC weights at the end of two stages by focusing the search without compromising on the exploration. There is no prior knowledge provided to the agents other than the action space given in (13).

Average number of desirable NMPC weights obtained by the agents is illustrated in Fig. 3. Since the complexity of the reward function increases from Agent$_{(1)}$ to Agent$_{(4)}$, the latter explores the least number of desirable NMPC weights. It needs to satisfy more criteria simultaneously which results in a lower quantitative performance. The same quantitative performance applies for all the agents proceeding from Stage 1 to 2. As depicted in Fig. 3, the number of the desirable NMPC weights decreases more than a half for each agent while moving from Stage 1 to 2. Since it is relatively easier for the quadrotor to hover as compared to moving to a particular setpoint, the agents satisfy their respective criteria more easily. In the same vein, they acquire a higher number of desirable weights and positive rewards accordingly, as depicted in Fig. 4. However, there is a trade-off between the number of the weight sets obtained and the flight performance with those sets as discussed in the next subsection.

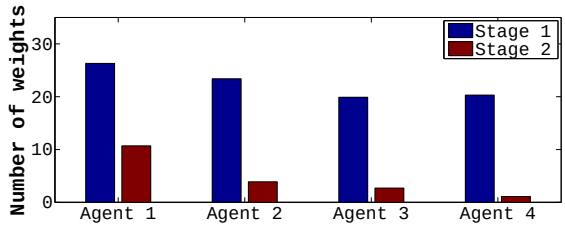

Fig. 3: Average number of desirable NMPC weight sets obtained by different agents.

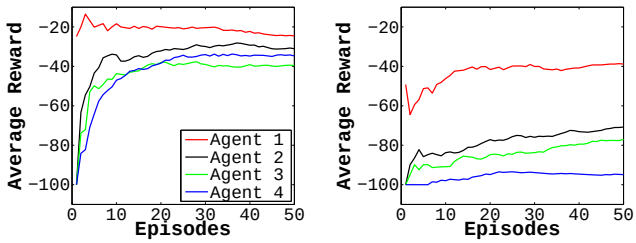

Fig. 4: Average rewards obtained by different agents in Stage 1 on the left, in Stage 2 on the right.

B. Benchmark

After exploring the NMPC weight sets by different agents, we deploy the quadrotor using these weights for a benchmark study. Unlike the previous subsection, here we present a qualitative discussion. The quadrotor follows a square-shaped trajectory with challenging jumps of 2m from one setpoint to another. We utilize all the weights obtained during learning for each agent, and assess the performance of them based on the averaged mean Euclidean error[1]. As can be seen in Fig. 5, Agent$_{(4)}$ yields the least averaged error over the trajectory. The NMPC weights explored by Agent$_{(4)}$ are competent on executing sudden jumps precisely, thanks to $R_{e_{ss}}$ component of its reward function. Some NMPC weight sets explored by the other agents are also able to yield desirable performance but degraded performance is also present with the other weights obtained by those agents. The reason for these results is that, from Agent$_{(1)}$ to Agent$_{(4)}$, the assessment of the NMPC weights become more strict with added criteria. While Agent$_{(1)}$ explores the widest set of NMPC weights which includes both favorable and unfavorable weights for realistic flight conditions, Agent$_{(4)}$ searches only for the ones which satisfy all the four criteria simultaneously. Therefore, it is able to explore more reliable weights at the end of the training. This outcome is also depicted in Table IV, where we define a failure criteria for the quadrotor as an altitude drop of 1.2m and a horizontal deviation of 5m from the nominal path. In this vein, Agent$_{(4)}$ yields the safest weight sets without failure. All the other agents have some degree of

[1]In this application, we consider the tracking error to be the distance between the current position and the corresponding closest point on the trajectory.

TABLE IV: Failure percentages for different agents.

	Agent$_{(1)}$	**Agent$_{(2)}$**	**Agent$_{(3)}$**	**Agent$_{(4)}$**
Failure percentage	19.6%	10.3%	11.1%	0%

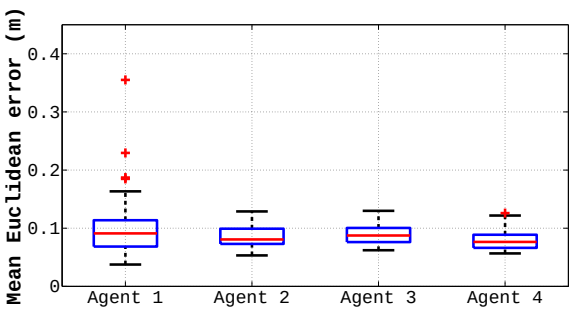

Fig. 5: Mean Euclidean error calculated using median for different agents.

failure. These results validate the reward function of $\text{Agent}_{(4)}$ for exploring reliable NMPC weight sets.

V. Experiments

After selecting the most desirable reward function for training, we utilize the NMPC weights obtained over a single run of learning with $\text{Agent}_{(4)}$ for a real flight in OptiTrack Lab at Nanyang Technological University. Similar to the benchmark study in the previous section, we deploy the quadrotor for tracking a square-shaped trajectory with sudden jumps of 1m. The agent converges to two weight sets at the end of 2-stage learning with 50 episodes in each stage, within 40 minutes on a Dell Precision Tower 5810 desktop computer, where the two weight sets are (i) weight set 1: $W_{\text{x}} = \text{diag}(40.06, 39.3486, 64.2519, 0.8437, 0.8626, 0.7841)$, $W_{\text{u}} = \text{diag}(50.194, 45.5608, 39.916, 0.0393)$, and (ii) weight set 2: $W_{\text{x}} = \text{diag}(12.304, 13.5076, 44.929, 0.6657, 0.6759, 0.4969)$, $W_{\text{u}} = \text{diag}(13.069, 13.006, 50.455, 0.0788)$.

As can be seen from Fig. 6, both NMPC weight sets yield a satisfactory performance in terms of trajectory tracking. The deviation levels in the horizontal plane are in the range of a few centimeters for the second weight set. The low deviation in the altitude also demonstrates a sufficient tracking. Among the two weight sets, while the second one has better performance in precision, the first one yields faster maneuvers. At this point, it is user's decision to choose either of the weight sets based upon the specific needs of the application of interest. As demonstrated, our learning framework is able to provide a few desirable NMPC weight sets within 40 minutes of training without requiring any human effort to tune NMPC weights or any prior knowledge other than the orders of magnitude of the NMPC weights.

VI. Conclusion

In this paper, we present a novel, automated tuning framework for (N)MPC in order to decrease the time and effort to be put by the human user. The incorporated RL technique facilitates a generic and system-independent tuning process. As a case study, we tested the proposed method for tuning of NMPC which is employed for position tracking of quadrotor UAV. The results have validated the computational efficiency of the proposed method as it obtains the desirable weights within less than an hour from the iterative Gazebo simulations running in a standard desktop computer. What is more, the real world experiments have demonstrated the direct applicability of the weights obtained from simulation while resulting in a satisfactory trajectory tracking performance.

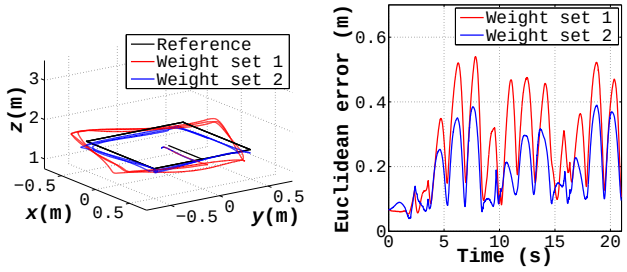

Fig. 6: Square-shaped trajectory tracking results on the left, Euclidean error on the right.

Acknowledgment

This work was financially supported by the Singapore Ministry of Education (RG185/17) and Aarhus University, Department of Engineering (28173).

References

[1] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4692–4697.

[2] C. Papachristos, K. Alexis, and A. Tzes, "Dual–Authority Thrust–Vectoring of a Tri–TiltRotor employing Model Predictive Control," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 3, pp. 471–504, 2016.

[3] H. Seo, S. Kim, and H. J. Kim, "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6362–6368.

[4] M. Mehndiratta and E. Kayacan, "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*, 2017.

[5] M. Mehndiratta, E. Kayacan, S. Patel, E. Kayacan, and G. Chowdhary, "Learning-based fast nonlinear model predictive control for custom-made 3D printed ground and aerial robots," *Control Engineering*, 2019.

[6] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açikmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.

[7] J. Lee and Z. Yu, "Tuning of model predictive controllers for robust performance," *Computers & Chemical Engineering*, vol. 18, no. 1, pp. 15 – 37, 1994, an International Journal of Computer Applications in Chemical Engineering.

[8] R. Shridhar and D. J. Cooper, "A tuning strategy for unconstrained multivariable model predictive control," *Industrial & Engineering Chemistry Research*, vol. 37, no. 10, pp. 4003–4016, 1998.

[9] A. Al-Ghazzawi, E. Ali, A. Nouh, and E. Zafiriou, "On-line tuning strategy for model predictive controllers," *Journal of Process Control*, vol. 11, no. 3, pp. 265 – 284, 2001.

[10] E. Ali, "Heuristic on-line tuning for nonlinear model predictive controllers using fuzzy logic," *Journal of Process Control*, vol. 13, no. 5, pp. 383 – 396, 2003.

[11] K. M. Cabral, S. R. B. dos Santos, S. N. Givigi, and C. L. Nascimento, "Design of model predictive control via learning automata for a single UAV load transportation," in *2017 Annual IEEE International Systems Conference (SysCon)*, April 2017, pp. 1–7.

[12] P. T. Jardine, S. N. Givigi, and S. Yousefi, "Experimental results for autonomous model-predictive trajectory planning tuned with machine learning," in *2017 Annual IEEE International Systems Conference (SysCon)*, April 2017, pp. 1–7.

[13] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," PhD dissertation, EPFL, 2007.

[14] B. Li, W. Zhou, J. Sun, C. Wen, and C. Chen, "Model predictive control for path tracking of a VTOL tailsitter UAV in an HIL simulation environment," in *2018 AIAA Modeling and Simulation Technologies Conference*. American Institute of Aeronautics and Astronautics, 8-12 January 2018, p. 1919.

[15] M. Vukov, S. Gros, G. Horn, G. Frison, K. Geebelen, J. Jørgensen, J. Swevers, and M. Diehl, "Real-time nonlinear MPC and MHE for a large-scale mechatronic application," *Control Engineering Practice*, vol. 45, pp. 64 – 78, 2015.

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

[17] 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.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.