Autonomous UAV with Learned Trajectory Generation and Control

Yilan Li[1], Mingyang Li[1], Amit Sanyal[1], Yanzhi Wang[2], Qinru Qiu[1]
[1]Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244, USA
{yli41, mli170, aksanyal, qiqiu} @syr.edu
[2]Department of Electrical and Computer Engineering, Northeastern University, MA 02115, USA
yanz.wang@northeastern.edu

***Abstract*—Unmanned aerial vehicle (UAV) technology is a rapidly growing field with tremendous opportunities for research and applications. To achieve true autonomy for UAVs in the absence of remote control, external navigation aids like global navigation satellite systems and radar systems, a minimum energy trajectory planning that considers obstacle avoidance and stability control will be the key. Although this can be formulated as a constrained optimization problem, due to the complicated non-linear relationships between UAV trajectory and thrust control, it is almost impossible to be solved analytically. While deep reinforcement learning is known for its ability to provide model free optimization for complex system through learning, its state space, actions and reward functions must be designed carefully. This paper presents our vision of different layers of autonomy in a UAV system, and our effort in generating and tracking the trajectory both using deep reinforcement learning (DRL). The experimental results show that compared to conventional approaches, the learned trajectory will need 20% less control thrust and 18% less time to reach the target. Furthermore, using the control policy learning by DRL, the UAV will achieve 58.14% less position error and 21.77% less system power.**

***Index Terms*--Deep reinforcement learning, continuous trajectory tracking, actor-critic algorithm, unmanned aerial vehicles.**

I. INTRODUCTION

Unmanned aerial vehicle (UAV) technology is a rapidly growing field with tremendous opportunities for research and applications. According to PricewaterhouseCoopers LLP consulting group, the global market for commercial applications of the UAV technology will rise to as much as $127 billion by 2020 with more than 6,000% increase by the end of the decade [1]. Currently, there is no technology that integrates UAV in the National Airspace System (NAS) with complete airworthiness. According to Mark Bathrick, Director of the Office of Aviation Services (OAS) in the Department of the Interior (DOI), "The biggest challenge to safely integrating UAV into the national airspace – with the ability to fly beyond the line of sight of the operator – is developing a system that enables UAV to "sense and avoid" other stationary or moving objects. Sense and avoid systems rely on radar, onboard transponders, or other active means. They suffer the same vulnerabilities as the two–way radio links used to control most drones" [2]. This calls for true onboard autonomy in real time for UAVs in the absence of remote control, external navigation aids like global navigation satellite systems (GNSS) and radar systems. The nonlinearly stable guidance, navigation and control will solely rely on vision and other inputs from low cost and low power onboard sensors, and be achieved through real-time detection, perception and control algorithms running on onboard hardware.

The term "UAV" refers to wide range of systems, from airplane-sized combat drones to insect-sized micro-drones. In this project, we will focus on small UAVs with take-off weights 2~20 lbs. This is not only because such small UAVs are commercially available and have the most civilian usage, but also because they impose more stringent constraints on the size and energy dissipation of the onboard system. Without loss of generality, motivating applications are remote monitoring and observation of a species in the wild, possibly under forest cover, reconnaissance missions, and search and rescue operations.

From autonomous obstacle avoidance to mission planning incorporating machine intelligence, we divide the autonomy of a UAV into four levels shown in Fig. 1. The lowest level is the reactive control (RC), which monitors the gyroscope and accelerometer data, and dynamically adjusts the motor to react to the changing airflow. The reactive control is important to stabilize the drone in a changing environment. The second level is trajectory generation (TG), where the drone dynamically calculates a sequence of waypoints and the trajectory that leads to a (changing) destination with the consideration of its current speed, acceleration, flight condition and potential obstacles. The third level is context aware adaptation (CAA), where the drone perceives its environment and adapts to a flight pattern or operation mode that gives the highest efficiency and safety. Image/video processing algorithms are usually employed in this layer across multiple modalities to accurately detect targets, objects of interest or obstacles. Based on the detection results, optimal decisions on flight and sensor control are searched and forwarded to the TG layer. The top level is strategic mission planning (SMP). At this level, the long-term mission goal is considered together with the availability of local resources, such as the remaining battery energy, to calculate a sequence of short term objectives and constraints that guide the lower level optimization and lead to best return. The local resources may include the status of neighbor drones if in a swarm environment.

This work is partially supported by the National Science Foundation under Grant CNS-1739748

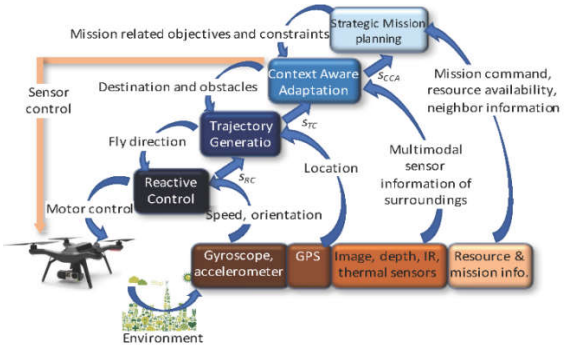


Figure 1. The four nested control loops in an autonomous UAV.

As shown in Fig. 1, the four layers of autonomy form four nested control loops. From RC to SMP, the layers process different sensory signals with an increasing complexity. The goal of each control layer is to generate a set of objective and constraint functions for the lower layer based on its sensor inputs with the consideration of the objectives and constraints that itself receives from the upper layer. For instance, given the image of an occluded object, the CAA controller will decide whether to move to a different angle to have a better view or to continue in the current direction depending on overall mission being searching specifically for a certain target and tracking its behavior or performing general surveillance. The decision and new destination will be sent to the TG layer and the new trajectory will be generated accordingly.

The two lower layers (i.e. RC and TG layers) work at the highest abstraction level (i.e. altitude, speed, etc.) with the least amount of information, however require the fastest response. The two upper layers (CAA and SMP) of system have relatively more relaxed constraints with respect to response time. Yet, they operate at the lowest abstraction level and deal with the most complex inputs and very large amounts of data (i.e. images, videos or IR scan images). Being able to process multiple channels of sensor inputs at higher granularity at each layer of the control loop, means safer and more efficient operation. The key enabling factor of the autonomous UAV is the capability to perform optimization, information fusion, detection and decision making rapidly in real time. In addition to computing hardware, new computing models and software framework need to be developed to enable multimodal sensor fusion, hierarchical perception, decision making and optimization.

The existing hardware and software solutions are not adequate to meet the requirements of fully autonomous UAV with real-time performance. First of all, the overall control optimization problem, especially the upper two layers in Fig. 1, has an extremely large solution space. A UAV needs to decide when to be at which location, and what type of sensing to perform and which data to collect in order to maximize the return of a particular mission while satisfying the resource and safety constraints. The typical mission duration is around 20 minutes and the range is up to 4~5 miles. There is no immediate rewards or penalties for the actions taken by the UAV. A final (delayed) reward/penalty is usually received at the end when the mission goal is accomplished/failed. Therefore, traditional reactive or predictive control cannot be applied to solve this problem. On the other hand, the emerging deep reinforcement learning (DRL) technique is data-driven and can potentially solve complicated control problems with extremely large state and action spaces, as well as continuous state space [3]-[5]. This technique could be potentially used for control policy optimization in the CAA and SMP layers.

In this work, we will focus on the CAA and SMP layers and discuss how their function can be learned and implemented using deep neural network models.

II. Learning Efficient Trajectory Generation

A. Problem Formulation

In this part, we focus on trajectory generation for multi-rotor UAVs. These UAVs have fixed plane of rotors that actuate the vehicle in three-dimensional transnational and rotational motion, hence they have the property of under-actuation. Given a closed environment, the UAV takes off from an arbitrary position and reaches a target position which is preassigned, without colliding with obstacles. As stated before, the first step is to select waypoints based on the environment. The entire 3D environment is divided into $N \times N \times N$ grids. The environment is described by a function $\mathbf{M}$ () maps a grid (x, y, z) to a real value $\boldsymbol{M}$: $(x, y, z) \rightarrow \boldsymbol{R}$. A grid, g, that contains obstacle will be mapped to -10, $\boldsymbol{M}(\mathrm{g}) = -10$. The destination grid that the agent needs to reach is mapped to 10, and the grid where the UAV is currently located is mapped to 1. All other grids are mapped to zeros in the discretized environment block. Let $W_0, W_1, \ldots, W_{N-1}$ be the sequence of generated waypoints, where each one is a 3D vector corresponding to a grid in the environment. Let $f(W_i, W_j)$ denote the control thrust for the UAV to follow the trajectory between waypoints W_i andW_j generated by the lower level optimizer. Also, let $\boldsymbol{G}(W_i, W_j)$ denote the set of grids that the generated trajectory between W_i and W_j will pass through. The total thrust cost along the trajectory is denoted as $\boldsymbol{F}$. The problem of waypoints generation can be formulated as the following:

PROBLEM 1 (Optimal Obstacle Avoidance Waypoints Planning). *Minimize*

$$\boldsymbol{F} = \sum_{i=0}^{N-2} f(W_i, W_j) \quad (1)$$

subject to

(1) *reaching the target position from current position*,

$$\boldsymbol{M}(W_0) = 1, \boldsymbol{M}(W_{N-1}) = 10, \quad (2)$$

(2) *reaching the target position without colliding with obstacles,*

$$M(g) \neq -10, g \in G(W_i, W_{i+1}),\ 0 \leq i \leq N-2 \quad (3)$$

To find the set of W_i, $0 \leq i \leq N-1$ is a combinatorial problem. The goal is to achieve minimum control thrust without obstacle collision if the UAV flies along the waypoints and trajectory. A large reward will be received at the end of the flight if the UAV reaches the destination. While this is the problem formulation of the upper level optimizer, the functions $f(W_i, W_j)$ and $\boldsymbol{G}(W_i, W_j)$ are determined by the lower level optimizer. Reinforcement learning provides a way to solve such constrained optimization problem with delayed reward.

Incorporating with deep neural network, an optimal policy is learned to guide the UAV to the next selected actions (i.e. waypoints) that can lead to maximum future rewards.

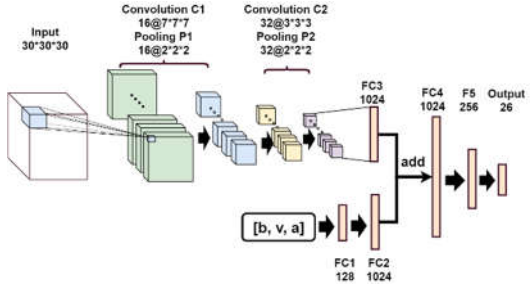


Figure 2. Network structure of proposed deep Q network.

B. Network Structure

A deep Q network is used to solve the problem and the detailed structure of the model is shown in Fig. 2. The inputs contain two parts. First, a 3D matrix with size $N \times N \times N$ is used to represent the current known knowledge of the surroundings. Each entry (x, y, z) of the matrix is the mapped value $\boldsymbol{M}(x, y, z)$ of the corresponding grid in the 3D environment previously discussed. It has the information about the relative position between agent and obstacles. In addition, a 1×9 vector is used to indicate the dynamic configuration of the UAV, where b, v, a represent the current position, current velocity and current acceleration of the UAV respectively. At each decision episode, a UAV can choose any of the $3 \times 3 \times 3$ grids around its current location as the waypoint. Therefore, there are 26 possible actions. The matrix input is fed into two 3D convolutional layers and each is followed by a pooling layer. The intermediate output of the second pooling layer is fed into a fully-connected layers with the size 1024. The configuration vector input is fed into two fully-connected layers. The intermediate outputs of FC2 and FC3 are fed into two fully-connected layers with the size 1024 and 256 respectively. The output is a fully-connected layer with the size 26. Each output neuron estimates the $Q(state, action)$ values for one of the 26 actions at the given state.

Our goal is to generate trajectory for the UAV with minimum control thrust under the premise of reaching target position without hitting any obstacle. Therefore, our reward function is defined as the combination of position reward and control reward as following:

$$\boldsymbol{R}(state, action) = \alpha R_p(state, action) + \beta R_c(state, action) \quad (4)$$

where α and β are the coefficients of position reward and control reward respectively, α = β = 0.5 in our experiment. $R_p(state, action)$ is the position reward of taking action in current state. It is defined as following:

$$R_p(state, action) = \begin{cases} 10 & \text{reach target position} \\ -10 & \text{collide with obstacles} \\ 0 & \text{others} \end{cases} \quad (5)$$

And $R_c(state, action)$ indicates the control reward. It is calculated as the negative L1 norm of thrust cost, which is calculated by LQR trajectory generation scheme proposed in [7].

C. Learning of DQN

Since the problem complexity, i.e. the total number of state action pairs, is $O(26 \times N^3)$ which is relatively larger than many other existing problems [8]-[10], it is crucial to maximize exploration at the beginning of learning. Therefore ϵ-greedy [17] is applied during the learning. Based on ϵ-greedy, more random actions (i.e. exploration) are taken at the beginning of learning and more actions with maximum $Q(state, action)$ values (i.e. exploitation) are chosen as learning progresses.

To improve the learning, we also decrease the learning rate lr gradually because it becomes harder to improve performance with large learning rate as the gradient reaches plateau. In our approach, there are 30,000 learning episodes in total. Instead of using a fixed learning rate, the learning rate starts from 1e-4 and decreases every 5,000 episodes based on $lr_{new} = (lr - \frac{lr}{epoch})_{epoch=i}, i \epsilon \{5e+3, 1e+4, 1.5e+4, 2e+4, 2.5e+4\}$ and i is the i_{th} learning episode. This helps to prevent the learning from over correct after 5,000 iterations, which allows to maximize the exploitation.

Instead of randomly initializing model weights based on a uniform distribution, we initialize the weights of model based on a normal distribution. It initializes weights with relatively small values and prevents outputs of the model from being either too large or too small. Batch normalization is used before the second convolutional layer and the first fully-connected layer to reshape the input of those hidden layers. In order to bound the training time, for every single training episode, the maximum steps W_{max} that the UAV can take is fixed. If the UAV has taken W_{max} steps but has not reached the target position, it will be forced to start a new learning episode. The start and target positions are randomly selected. So do the locations of obstacles. In this way the environment configuration of every episode is different, therefore each learning episode is independent. During learning, an experience replay is used to save last thousand times of performance and a randomly sampled mini-batch of size 32 is used to train the network. At each time step within an episode, the UAV takes an action and receives a +10 position reward if it reaches the target position and -10 if colliding with obstacles. Otherwise the position reward is 0. The control reward is determined based on the current and next position, velocity and acceleration of the UAV. The weighted sum of these two rewards and corresponding UAV state and action is saved in experience replay buffer.

D. Gain Selection through Genetic Algorithm

The control thrust is the feedback reward for higher level waypoints planner during learning. It is necessary to mention that in the lower level of our module, LQR trajectory generation scheme, Q, R and S are three positive definite gain matrices. Each of these gain matrices penalize different aspects while generating the trajectory. Q is a matrix penalizing high values of position, velocity and acceleration. The higher the values in Q, the harder these parameters are penalized. The input jerk is penalized when R matrix has large eigenvalues and how much the waypoints will affect the trajectory is weighted by S matrix. We can see that these three gain matrices have significant impact on the performance index J^d of the trajectory. Their values need to be tuned in order to minimize the control thrust f. In our experiment, f is calculated from the lower level scheme, while it will be measured by sensors in real field learning. We apply genetic algorithm (GA) [13] to optimize the gain matrices. The best set of (R, Q, S) satisfies $\min_{R,Q,S} f(W_i, W_{end}, v_i, a_i)$ where W_i, v_i, a_i are initial position, velocity and acceleration of the UAV and W_{end} are the destination for UAV.

III. Learning Accurate Trajectory Control

A. Problem Definition

In this section, we consider the trajectory tracking for under-actuated aerial vehicles through a set of given desired way points. We aim at quadrotor fixed-wing UAVs with four control inputs, one degree of translation motion and three degrees of rotation motion (i.e. pitch, roll and yaw.) It is extremely difficult to integrate detailed mechanistic model of complicated dynamics with classic control theory, a model-free solution for the control problem is preferred. Because actions of the tracking problem are continuous variables (e.g. turning force, thrust etc.), the actor-critic reinforcement learning is adopted, which learns to find the optimal set of actuations that move the UAV towards desired trajectory. The technique presented in [11] is used to generate C^2 trajectory based on a given set of predefined waypoints T_d where each waypoint gives the desired position of the UAV at time t. Meanwhile, desired velocity vd_t, desired acceleration dvd_t and desired attitude Rd_t are extracted from T_d based on kinematics. The positions and attitudes together form the pose of the UAV. The goal of our model is to minimize the differences between desired poses and actual poses during tracking. We define $sd_t = \{pd_t, vd_t, dvd_t, Rd_t\}$ as desired state and $s_t = \{p_t, v_t, dv_t, R_t\}$ as actual state of UAV at time step t. Each of first three variables in the sd_t and s_t are 3-dimensional vectors and the last one in the sd_t and s_t is 3x3-dimension, hence the desired state and the actual state are variables in 18- dimensional space.

The concatenation of sd_t and s_t forms the agent state S*t*. Furthermore, we define action at time t as $A_t = \{fm_t;\ \tau_t\}$, where the translational force fm_t and torques τ_t are applied to UAV. The translational force fm_t is a force perpendicular to the top surface of UAV and the three components in τ_t are applied to roll, pitch and yaw directions respectively. The reward $\mathrm{Reward}(\Delta t)$ at time t is defined as the Manhattan distance between the desired pose and the actual pose, i.e. $\mathrm{Reward}(\Delta t) = f(|pt - pdt| + |vt - vdt| + |Rt - Rdt|)$.

B. Network Structure

Since the control variables (i.e. fm_t, τ_t) of UAV are in a continuous space which are infinitely large, we build an actor-critic reinforcement learning model instead of discretizing the action space. The actor model is a feed-forward deep neural network of three fully-connected hidden layers with rectified linear units (ReLU) as the activation function. It is used to predict the optimal action based on current state S_t. The number of neurons in fully-connected hidden layers are 64, 128 and 128 respectively. The size of output layer is 4 and LeakyReLU activation function is used in the output layer. The critic model is another feed-forward neural network that computes an evaluation of the action and that evaluation is used by actor model to update its control policy in particular gradient direction. The critic model has two hidden layers, where the first layer contains two separate fully-connected structures and the number of hidden neurons in each is 32. The addition of outputs from the first hidden layer is fed into the second layer which has 64 hidden neurons. The inputs of critic model are S_t and A_t and the output is a single value $Q(S_t, A_t)$. The size of the critic and actor is optimized as hyper-parameters through cross-validation. The detailed networks of actor model and critic model are shown in Fig. 3a and Fig. 3b. The overall framework is shown in Fig. 3c. During training, the actor model is pre-trained using labeled pair data (S_t, A_t) generated from simulation [11] to predict the optimal action A_t based on current agent state S_t. Next agent state S_{t+1} is calculated through environment simulation based on A_t and is used to predict optimal A_{t+1} by actor model. The critic model evaluates the resulting $\{S_{t+1}, A_{t+1}\}$ pair by predicting a Q-value to fine-tune action prediction. Therefore, the weights in actor model are updated by the gradient between actor and critic model, using chain rule $\frac{dQ}{dW_{actor}} = \frac{dQ}{dW_{critic}} \times \frac{dW_{critic}}{dW_{actor}}$. dW_{actor} and dW_{critic} indicate the weights of actor and critic models respectively.

C. Reward Definition

Our goal is to actuate the UAV to be closer to desired pose T_d along the desired trajectory, i.e. minimizing the value of $\Delta \mathrm{P_t} = |p_t - pd_t|$. Besides position error, the stability of the UAV should also be taken into consideration. Therefore the values of velocity errors (i.e. $\Delta \mathrm{V_t} = |v_t - vd_t|$.) and attitude errors ($\Delta \mathrm{R_t} = |R_t - Rd_t|$.) must also be minimized. However, our experiments show that simply using a linear combination of $\Delta \mathrm{P_t}$, $\Delta \mathrm{V_t}$, and $\Delta \mathrm{R_t}$ as the reward function will make convergence difficult in learning process. According to [12], using geometrically discounted reward will prevent the accumulated reward to become infinite and make the model more tractable. Therefore, we define the reward at each time step following a standard normal distribution, guaranteeing the largest reward is accepted when the total differences between desired trajectory and actual trajectory at time t (i.e. $\Delta \mathrm{t} = \Delta \mathrm{P_t} + \Delta \mathrm{V_t} + \Delta \mathrm{R_t}$) reaches zero and closing to zero reward is obtained when Δt increases. The total discounted reward is denoted as R.

$$Reward(\Delta t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\Delta_t^2}{2}\right)$$
$$R = \sum_{t=0}^{\infty} \gamma^t Reward(\Delta t) \quad (6)$$

where $\Delta \mathrm{t} = \Delta \mathrm{P_t} + \Delta \mathrm{V_t} + \Delta \mathrm{R_t}$.

IV. Experimental Results

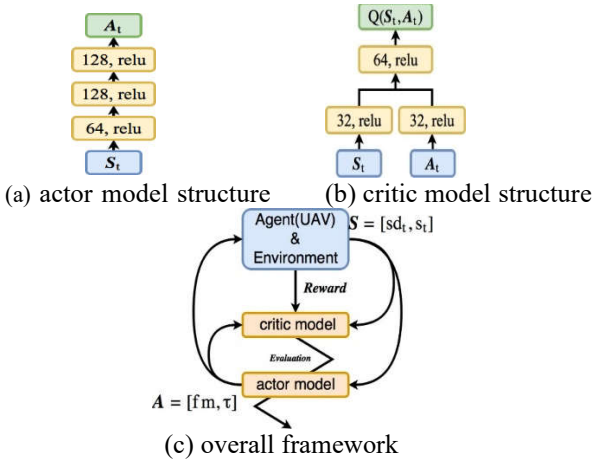
In this section, we demonstrate the performance of our proposed model. The training and testing were done on NVIDIA TitanX (Pascal). In the experiments, the environment is divided into $10 \times 10 \times 10$ and the unit distance δ_d is 10 meters. Within each testing scenario, the number of obstacles is randomly generated, and obstacles are placed randomly within the environment boundary. Besides, the start and target positions are also randomly selected. We report the results compared with some existing approaches.

A. Results Comparison for Trajectory Generation

As a baseline of reference, we also implement a control scheme based on PID theory [14] to estimate the control thrust consumption along waypoints and use it to replace LQR trajectory generation scheme in [7] as the lower level optimizer. We define position as measurable variables and velocities when reaching each waypoints are regarded as controllable variables. Every time the position of the agent is updated by the feedback of the environment, and the feedback is calculated by the environment simulation based on Kinematic theory [15].

We compare the DRL based waypoints selection with four traditional waypoints selection approaches in aspects of average number of steps needed to reach target, and the average control

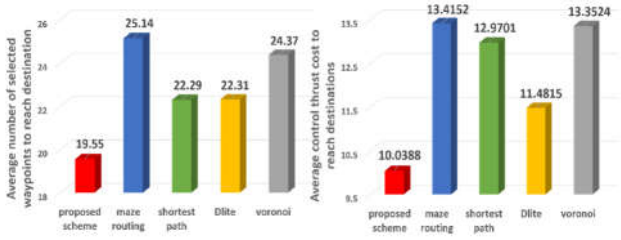
thrust cost along the trajectory. These four approaches include maze routing, shortest path, DLite algorithm and voronoi path. To make it more convincing, the size of discretized environment is set as $30 \times 30 \times 30$. We generated 1000 different test scenarios by randomly select different start positions, destination positions, types and locations of obstacles. Fig. 4 compares the average number of selected waypoints to reach destination and the average control thrust cost for the UAV to go through these waypoints. For all waypoints selection approaches, LQR trajectory generation scheme in [7] is used for trajectory generation. In Fig. 4a, the results show that our approach only needs an average 19.55 waypoints which is 16.91% less than other approaches. In Fig. 4b, the comparison of average control thrust consumption is reported. As indicated in the figure, our approach uses least control thrust which has 18.87% reduction than other approaches. According to these results, our purposed scheme with fewer selected waypoints is less possible to be over constrained and less times of lower level scheme invocation is needed.



(a) actor model structure (b) critic model structure

(c) overall framework

Figure 3. Architecture of actor and critic model, and framework.



(a) Average number of waypoints (b) Average control thrust

Figure 4. Comparison with existing approaches proposed DQN scheme, routing, shortest path, DLite and Voronoi.

B. *Environment Setup for Actor-critic Network*

We trained the actor-critic network and implemented the simulation on Nvidia GeForce GTX1070 using Keras [16]. The data that we use to train and test our model is generated using simulator described in [11]. Our dataset consists one thousand different 3-D trajectories of four different shapes, including straight lines, z-shape curves, spiral curves and circles. Each desired trajectory has one thousand desired waypoints, giving enough time for UAV to track it. All desired waypoints are defined by mathematical equations parameterized by time. Eight hundred different trajectories of four different shapes are evaluated in total. The mass of UAV used is 4.34kg, and its inertial properties J is a 3x3 diagonal matrix (i.e. diag[0.820 0.0845 0.1377]kgm^2) which determines the required magnitude of force to accelerate the UAV in each rotation direction respectively. The goal is to minimize the power consumption and time used from trajectory deviated to tracked.

C. *Results Comparison on Tracking Aspect*

All test samples of desired trajectories are generated the same way. We report the results from four aspects: (1) L1- norm of position tracking error; (2) L1-norm of velocity tracking error; (3) Time used to complete tracking; (4) Power consumption.

Fig. 5 shows the L1-norm of position tracking error, where the first four columns are comparison results of trajectories with different shapes respectively and the fifth column is average L1-norm of position error of all testing trajectories. According to the figure, our approach has lower average position error especially when the trajectory is more complicated.

Compared to PID based baseline control, 22.94% less position error is achieved for straight line trajectory tracking and 25.07% less position error is achieved for circular trajectory tracking. The position tracking error reduction increases to 85.32% for spiral shape tracking and 87.81% for z-shape trajectories tracking respectively. On average, our approach outperforms 58.14% better than PID based control in position tracking of different shapes of trajectories.

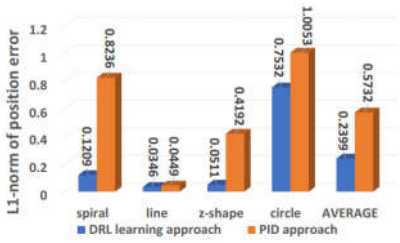


Figure 5. Tracking result comparison between proposed DRL- based framework and PID-based baseline in terms of L1-norm of position error.

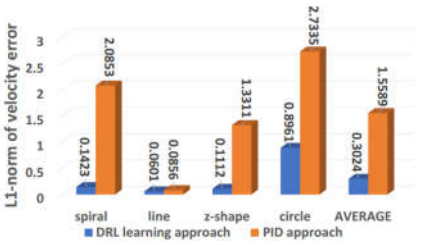


Figure 6. Tracking result comparison between proposed DRL- based framework and PID-based baseline in terms of L1-norm of velocity error.

Fig. 6 shows the average of L1-norm of velocity tracking error. Similar to Fig. 5, the first four columns are comparison results of each type of trajectory with different shapes, and the last column shows trajectories. It shows 29.79% error reduction is achieved for line trajectory and 67.22% error reduction is achieved for circle trajectory. Up to 91.64% and 93.17% error reductions are achieved for z-shape and spiral trajectories. On average, our approach outperforms 58.15% than PID control with respect to velocity tracking error. Again, our learning-based approach performs better for the more complicated trajectories.

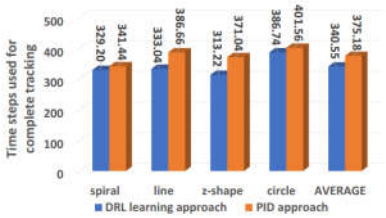


Figure 7. Tracking result comparison between proposed DRL-based framework and PID-based baseline in terms of used tracking time steps.

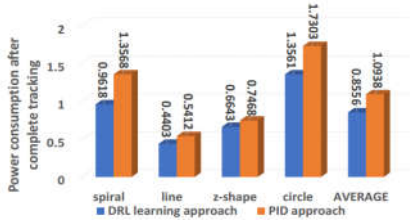


Figure 8. Tracking result comparison between proposed DRL- based framework and PID-based baseline in terms of power consumption.

Fig. 7 compares the total time steps used for UAV to follow each shape of desired trajectory and the average time steps used in total to reach stability. We report the number of time steps used for UAV to completely track the desired trajectory. The total time steps for each testing trajectory is one thousand. The time step when trajectory is tracked is regarded as the time t_c after which the L1-norm of position error between pd_t and p_t is less than 0.0001. The average tracking time for different types of trajectories, and the average tracking time over all testing trajectories are reported. Our approach is 9.23% faster than PID-based control to achieve stable pose on average. It is especially 13.86% faster for line trajectory and up to 15.58% faster for z-shape trajectory. Moreover, PID has lags in responding current dynamics in all three directions of velocity. Therefore, PID-based controller is not optimally adapted for non-linearity situations, especially not robust in fast dynamic control. It trades off the control performance and time.

Fig. 8 reports average total power consumption after one trajectory is completely tracked using our approach and baseline PID controller. As indicated in the figure, our approach achieves 11.04% less power consumption when tracking shape trajectory and consumes 18.64% less power for line trajectory tracking. Furthermore, up to 21.63% and 29.11% power consumption improvements have been achieved for circle and spiral trajectories tracking respectively. An average of 21.77% less power is consumed using our approach for tracking all different trajectories. The noticeable result explains that PID control consumes more power because of oscillations of controllable variables during tracking process.

V. CONLUSIONS

A two-level framework to generate navigation trajectory for UAVs to follow in a complex environment is introduced. The framework's construction, processing and analysis are presented. The proposed waypoints planning framework effectively avoids obstacles in complex indoor environment and reduces the control thrust consumption during flight. Also, it is general enough to be applied in other robotics tasks such as parcel delivery and conflicting routing of high-density UAVs.

We also introduced a framework for UAV trajectory tracking based on deep reinforcement learning. The system structure, processing algorithm and software/hardware performance are presented. In our approach, the UAV tracks a desired trajectory through a set of given waypoints, tolerating random Gaussian noise within considerable range. The hardware consumption of the implementation of this scheme is also provided. The proposed scheme is general and applicable to be applied in real UAVs for fast and accurate trajectory tracking system.

REFERENCES

[1] "Global Market for Commercial Applications of Drone Technology Valued at over $127 bn," PWC, May 2016. [Online]. Available: http://press.pwc.com/News-releases. [Accessed March 2017].

[2] H. Swan, "UNMANNED AIRCRAFT SYSTEMS: GREAT CHALLENGES; GREAT POSSIBILITIES," StartupGrind, 2015. [Online]. Available: https://www.startupgrind.com/blog/unmanned-aircraft-systems-great-challenges-great-possibilities/. [Accessed March 2017].

[3] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in Advances in Neural Information Processing Systems, 2015.

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

[5] G. Dulac-Arnold, R. Evans, H. v. Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris and B. Coppin, "Deep reinforcement learning in large discrete action spaces," arXiv:1512.07679, 2015.

[6] Q. Chen and Q. Qiu, "Enhancing bidirectional association between deep image representations and loosely correlated texts," in International Joint Conference on Neural Networks (IJCNN), 2016.

[7] Hossein Eslamiat, Yilan Li, Ningshan Wang, Sanyal Amit, and Qinru Qiu. 2019 Autonomous Waypoint Planning, Optimal Trajectory Generation and Nonlinear Tracking Control for Multi-rotor UAVs. Accepted by 2019 European Control Conference.

[8] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li FeiFei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 3357–3364.

[9] Juan Wu, Seabyuk Shin, Cheong-Gil Kim, and Shin-Dug Kim. 2017. Effective lazy training method for deep q-network in obstacle avoidance and path planning. In Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on. IEEE, 1799–1804.

[10] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. 2018. Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. In 2018 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 115–123.

[11] S. P. Viswanathan, A. K. Sanyal, and E. Samiei, "Integrated guidance and feedback control of underactuated robotics system in se (3)," Journal of Intelligent & Robotic Systems, 2018..

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of artificial intelligence research, vol. 4,pp. 237–285, 1996

[13] Agoston E Eiben, James E Smith, et al. 2003. Introduction to evolutionary computing. Vol. 53. Springer

[14] Lluis Pacheco and Ningsu Luo. 2015. Testing PID and MPC performance for mobile robot local path-following. International Journal of Advanced Robotic Systems 12, 11 (2015), 155.

[15] Sandeep Kumar Malu and Jharna Majumdar. 2014. Kinematics, localization and control of differential drive mobile robot. Global Journal of Research In Engineering (2014).

[16] F. Chollet et al., "Keras." https://github.com/fchollet/keras, 2015.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013.