2018 International Conference on Unmanned Aircraft Systems (ICUAS)
Dallas, TX, USA, June 12-15, 2018

Vision-Based Autonomous Landing of a Multi-Copter Unmanned Aerial Vehicle using Reinforcement Learning

Seongheon Lee
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
skynspace@kaist.ac.kr

Taemin Shim
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
tmshim@ascl.kaist.ac.kr

Sungjoong Kim
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
sjkim@ascl.kaist.ac.kr

Junwoo Park
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
jwpark@ascl.kaist.ac.kr

Kyungwoo Hong
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
kwhong@ascl.kaist.ac.kr

Hyochoong Bang
Department of Aerospace Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
hcbang@ascl.kaist.ac.kr

***Abstract*—This paper presents vision-based landing guidance of multi-copter Unmanned Aerial Vehicle (UAV) using reinforcement learning. In this approach, the guidance method is not designed or proposed by a human, but deployed by a neural network trained in simulated environments; which contains a quad-copter UAV model with Proportional-Integral-Derivative (PID) Controller, ground looking camera model that gives pixel deviation of targeting landing location from the center of an image frame, and laser rangefinder that gives altitude above ground level. Since we aimed for various types of multi-copter UAVs to track targeting ground location, reinforcement learning method has been used to generate proper roll and pitch attitude commands in multiple situations. Series of flight experiments show that a multi-copter UAV equipped with a proper attitude controller and trained artificial intelligence pilot can guide a multi-copter UAV to a ground target position.**

Keywords*—*autonomous landing, multi-copter, unmanned aerial vehicle (UAV), artificial intelligence (AI), reinforcement learning, neural networks

I. Introduction

The recent market demands on multi-copter Unmanned Aerial Vehicles (UAVs) are various in industry and military fields. Those demands, for instances, are aerial photography, farming, delivering, environment monitoring, asset monitoring, structure inspection, surveillance and reconnaissance, search and rescue, and so forth [1]. To develop a fully autonomous UAV, precise autonomous landing ability is a key to the mission completeness and using a vision-based system is a favorite solution due to its passive and low-cost characteristics [2]. By preparing a vision-based autonomous landing capability, one can fully utilize the capability of UAVs in various environments, such as GPS denied conditions, and cooperative missions with a moving ground vehicle or watercraft as a carrier [3, 4].

In the previous researches, autonomous landing of Vertical Takeoff and Landing (VTOL) UAV has been studied by applying computer vision algorithms and estimating the motion of a landing target position [5-7]. In our works, however, designing a fancier computer vision target-tracker; multi-copter UAV controller, such as Stability Augmentation System (SAS); or guidance formulas to enhance the autonomous landing performance are not the primary concern. The fundamental idea of this research is designing and training an Artificial Intelligence (AI) system that can maneuver a multi-copter UAV as if a human pilot who controls a multi-copter UAV through the control sticks of a radio transmitter. By doing so, we would like to introduce a guidance method that performs the same function with less effort.

For starter, we built a training environment with a quad-copter UAV dynamic and applied simple PID controller that can handle attitude and position of the UAV. Second, we observed the state from the environment and designed a proper reward function that our reinforcement learning algorithm can refer to. Finally, we trained a neural network, which has a policy network called the actor and a value network called the critic. The remainder of this paper is organized as follows. In Section II, training environments containing quad-copter dynamics, controller, and camera model are introduced. Section III starts from the fundamental concept of reinforcement learning, describes the actor-critic algorithm, vision-based autonomous landing framework, and shows some training results. In section IV, flight experiment results obtained from hexacopter UAV equipped with an off-the-shelf Naza flight controller and onboard graphics processing unit are presented.

II. Simulation Environments to be Trained

A. Dynamic Model of a Quad-copter UAV

Dynamic model of a quad-copter UAV can be built from [8], which also contains some physical properties of a small sized Micro Aerial Vehicle (MAV). The coordinate system and free body diagram are shown in Fig. 1.

Here, the coordinate systems are somewhat different from the common aerospace coordinate systems, where the world frame W takes the axes X_W, Y_W, and Z_W pointing upward and the body frame B takes the axes X_B pointing the nose, Y_B pointing the left wing, and Z_B pointing the upward direction originated from the center of mass. Therefore, the governing equation of translational motion is given by (1):

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \tag{1}$$

where R indicates the rotation matrix from body frame to world frame with the order of Z(yaw, ψ) – X(roll, ϕ) – Y(pitch, θ) Euler angles as follows.

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

where c() and $s()$ denotes cos() and sin() respectively.

The rotational motion of a quad-copter UAV with respect to the world frame can be determined by (3):

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3}$$

where p, q, and r are the angular rate of the UAV with respect to the body frame, which can be derived from the Euler angle measurements as (4); I is the 3×3 moment of inertia matrix with all zeros except $I_{xx} \approx I_{yy} = 0.0003\, kg \cdot m^2$; and $l = 0.1m$ stands for the distance between the center of mass and center of a motor.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi\, s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi\, c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{4}$$

Finally the motor dynamics can be expressed as (5)-(7) that a motor produces a vertical force and moment proportional to the square of angular speed, and the angular speed of the motor is approximated as a first order differential equation:

$$F_i = k_F \omega_i^2. \tag{5}$$

$$M_i = k_M \omega_i^2. \tag{6}$$

$$\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i). \tag{7}$$

where constant $k_F \approx 6.11 \times 10^{-8} N/(r/min^2)$, $k_M \approx 1.5 \times 10^{-9} N \cdot m/(r/min^2)$, motor gain $k_m \approx 20 s^{-1}$, and ω_i^{des} is a desired motor speed.

B. Controller of the Quad-copter UAV

The controller of the quad-copter UAV also can be found from [8] except the outer position control loop will be replaced by a trained artificial intelligence network, which will be introduced in the next chapter. The schematics of the inner loop controller are given in Fig. 2.

Here, the attitude controller interprets commands from pilot that are desired roll angle ϕ^{des}, pitch angle θ^{des}, and yaw angle ψ^{des} respectively; and gives additional motor commands $\Delta\omega_\phi$, $\Delta\omega_\theta$, and $\Delta\omega_\psi$ that will produce additional roll, pitch, and moments from the nominal state. $\Delta\omega_F$ affects net force increase or decrease along the Z_B axis from the nominal hover state motor speed ω_h. The desired motor speeds can be written as (8):

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \tag{8}$$

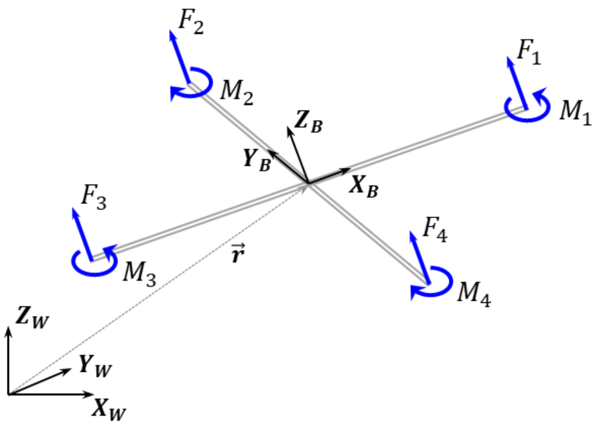

Fig. 1. Coordinate system and free body diagram of a quad-copter UAV.

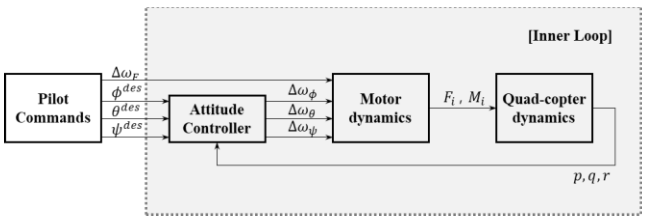

Fig. 2. Schematics of an inner-loop attitude controller of a quad-copter UAV.

C. Ground Looking Camera Model

Now the last thing to prepare is a ground looking camera model mounted on the gimbal assembly (distorted picture). Fig. 3 shows the schematics of a ground looking camera model. Here a gimbal assembly can control two axes, i.e., roll and pitch angles of the gimbal. In this configuration, the assembly controls the camera image frame to be aligned with the world frame that $\mathrm{Z_W}$ and $\mathrm{Z_I}$ are pointing the same direction regardless of the UAV attitude change.

The detailed description of the image frame I with respect to the world frame W is pictured in Fig. 4. In this diagram, optical axis Z_I is parallel with the Z_W axis that $X_W - Y_W$ plane, which is centered at the targeting landing location and $X_I - Y_I$ plane, which is centered at the digitized image frame are also parallel. Therefore, the difference of target center pixel with respect to the image center pixel can be used to control the position of multi-copter UAV in order to keep the targeting location within a camera field of view.

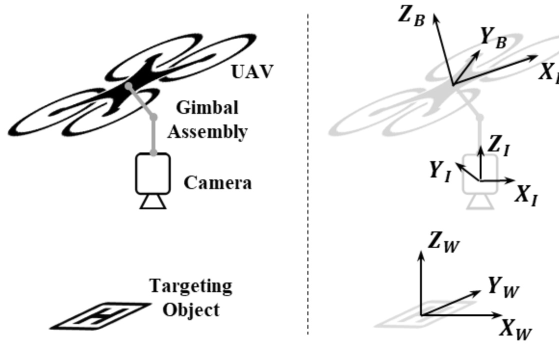

Fig. 3. Schematics and coordinate system of a ground looking camera model.

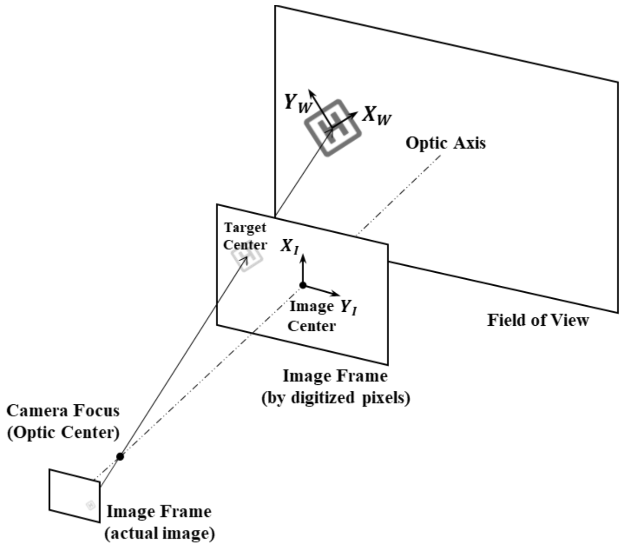
Y_W
X_W
Optic Axis
Target Center
X_I
Image Center
Y_I
Field of View
Image Frame (by digitized pixels)
Camera Focus (Optic Center)
Image Frame (actual image)

Fig. 4. Coordinate system of a image frame.

III. Reinforcement Learning Framework

Here we adopt Reinforcement Learning (RL) algorithm to substitute pilot commands, which give desired roll and pitch commands to the attitude controller. In RL schematics, we let our agent to get states, take actions, i.e., roll and pith commands from the given states, and get rewards from the simulation environments repeatedly. To be more specific, we trained a policy function called the actor, and a value function called the critic simultaneously to decide proper actions from given states by giving feedback to the actor to maximize the rewards.

Many RL algorithms can be found from [9], and deep RL algorithms that represent the policy function and/or value function as deep neural networks. This framework showed remarkable performance on various fields, such as Atari games [10], Go [11], and several dynamic simulation environments [12]. Since we are also dealing with the states and actions of UAV in continuous domain, deep RL approach can be applied to solve the problem.

A. Actor-Critic Framework for the Autonomous Landing

The overall schematics of an actor-critic framework for the vision-based autonomous landing is summarized in Fig. 5. When the training is completed, only actor activates when there is a need for the autonomous landing assistance to a human pilot. Altitude and heading control is not controlled by the agent, yet supposed to have constant speed descending with fixed heading direction. The detailed training procedures for deep reinforcement learning with the actor-critic algorithm can be found from [12]. Therefore, we only summarize states in use, desired actions, and reward function for the framework from (9) to (11).

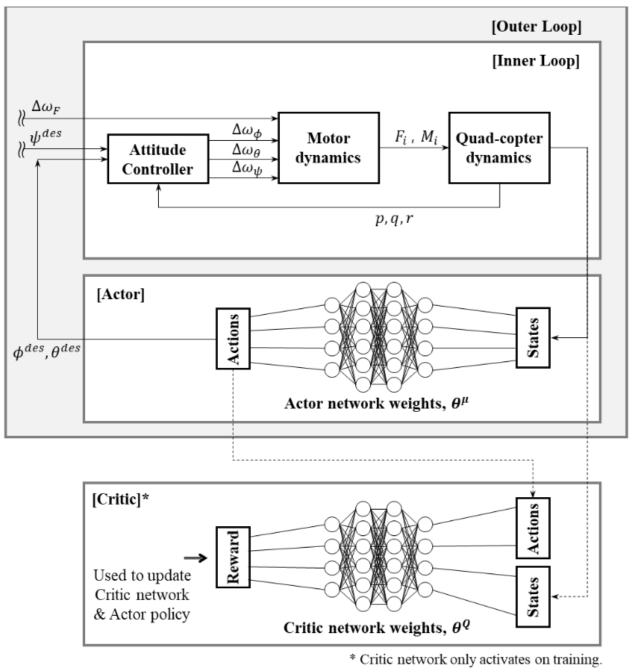

Fig. 5. Framework for the vision-based autonomous landing using actor-critic reinforcement learning.

The observed states from the UAV dynamics are as follows:

$$\vec{s_t} = \left[\, \tilde{h} \;\; \widetilde{dX_I} \;\; \widetilde{dY_I} \;\; \tilde{u} \;\; \tilde{v} \;\; \tilde{p} \;\; \tilde{q} \;\; \tilde{\phi} \;\; \tilde{\theta} \,\right]^{\mathrm{T}} \tag{9}$$

where h is the UAV altitude above ground level, dX_I and dY_I are difference between the center pixel of targeting landing-site on the image and center pixel of an image frame, u and v are body frame velocity on X_B and Y_B directions, p and q are angular rate of the UAV with respect to the X_B and Y_B, and finally ϕ and θ are roll and pitch Euler angles. Here the tilde sign '~' stands for the values are normalized from -1.0 to 1.0. This is a common technique to enhance the training performance.

Actions from the actor policy give two real-valued outputs ranging from -1.0 to 1.0, which are normalized desired roll and pitch attitude of UAV:

$$\vec{a_t} = \left[\, \tilde{\phi}^{des} \;\; \tilde{\theta}^{des} \,\right]^{\mathrm{T}} \tag{10}$$

The reward function that the agent keep try to maximize during the training period is as follows:

$$r_t = 1 - \left(\left(\frac{\widetilde{dX_I}}{a} \right)^2 + \left(\frac{\widetilde{dY_I}}{b} \right)^2 \right) \tag{11}$$

where $a = 0.6667$, and $b = 0.5000$, which are acting as semi-major axis and semi-minor axis of the zero-reward boundary (Fig. 6).

When the tracking landing-site is located at the center of the image frame, the agent takes +1 point, while the agent takes -5.2500 points when the landing-site is located at the each corner of the image. If the tracking landing-site is not within the field of view, the episode is over.

The actor network has two fully connected hidden layers with 300 weights each. The critic network also has two fully connected layers where the first hidden layer has 600 weights; 300 weights connect actions, while the other 300 connect states; and the second hidden layer has 300 weights fully connecting previous 600 weights. The activation function for the layer is ReLU except for the output layer of the actor network uses tanh since the output command should be lay between ±1. For the optimizer, we used ADAptive Moment estimation (Adam method) with α=0.001, β1=0.9, β2=0.999, and ε=10^{-8}.

The training environments built in the Section II updates the UAV states and targeting image location every one millisecond by using Runge-Kutta 4th order method, and the reinforcement learning agent learns the environment and the optimized actions from it. With a Graphics Processing Unit (GPU) of Nvidia GTX 1080Ti, the training takes only 6 hours to make an agent learn the environment over 3,000 episodes.

B. *Training Results*

The training has been done with three fully connected hidden layers with 300 weights each. Fig. 7 shows the performance of the actor during 3,000 repetitive training episodes. Part of the training results stores flight trajectories, and they are described in Fig. 8 to Fig. 10. In every episode, the simulated UAV starts from 10m height with the randomized initial position, velocity, and small external forces to compensate wind or unknown disturbances.

The results show that the neural network agent figured out how to maneuver the UAV within the observed states given in (9) with only two actions (10). The preferred landing location is located at the origin of each graph. In the early stages of learning episodes, exploratory behaviors such as moving away from the targeting location or flying near the target point can be observed. However, as the learning is continued, we can confirm that the UAV is directed toward the desired landing site.

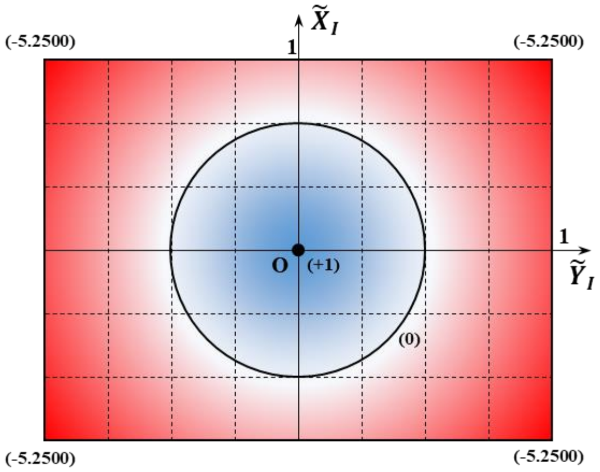

Fig. 6. Description of the reward function.

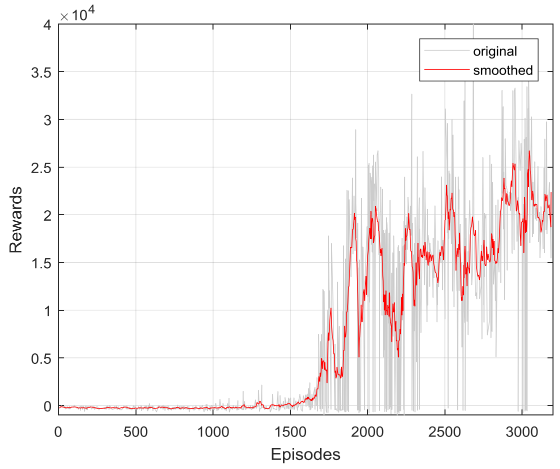

Fig. 7. Performance curve of the actor during 3000+ episodes.

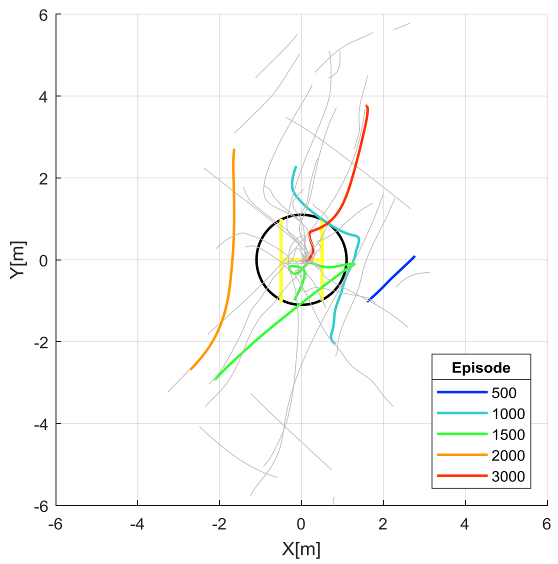

Fig. 8. Flight trajectories druing reinforcement learning (top view).

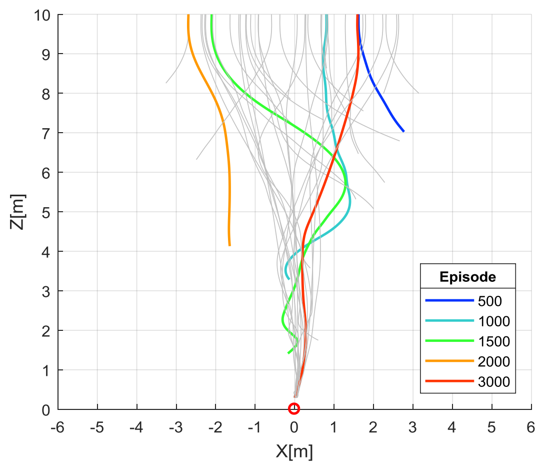

Fig. 9. Flight trajectories druing reinforcement learning (side view 1).

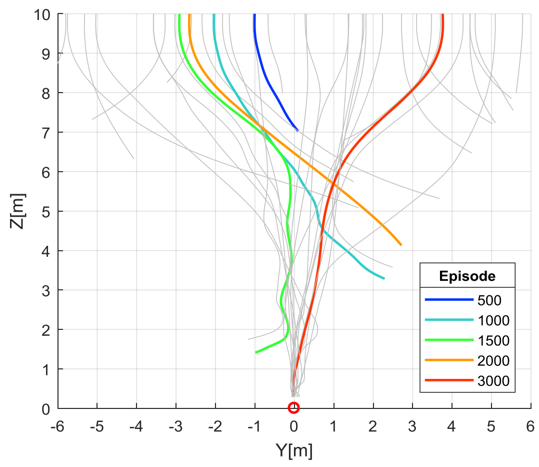

Fig. 10. Flight trajectories druing reinforcement learning (side view 2).

IV. Flight Experiments

A. System Descriptions

For the flight experiment, we used Tarot T810 hexa-copter frame to build a test vehicle. Table I. shows the overall specifications of the test vehicle. For inner loop attitude control, we used off-the-shelf Naza flight controller. Image processing for targeting landing-site tracking and parallel computations for the actor neural network processing are done by onboard Graphics Processing Unit (GPU). Secondary flight control computer takes outer loop for altitude and heading control and interprets actions from the GPU to give desired roll and pitch command to the inner loop controller.

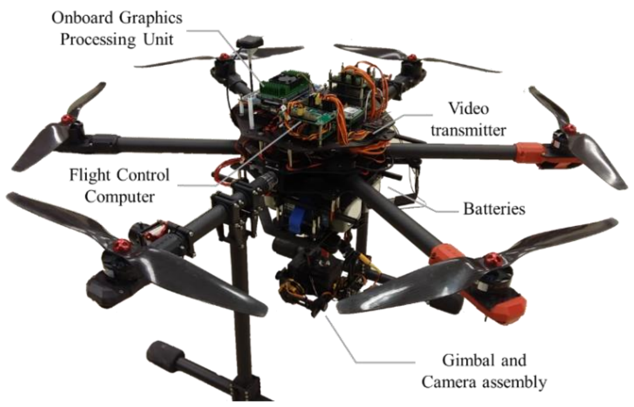

Fig. 11. Configurations of the experimental vehicle.

TABLE I. Experimental UAV System Specifications

	Specifications
Dimensions (W x H, mm)	800 x 500
Empty Weight	3.5 kg
Maximum Take-off Weight	12.9 kg
Flight Time	7 mins (with 9.2 kg take-off weight)
Flight Control Computer (Inner loop controller)	DJI NAZA-M V2
Flight Control Computer (Outer loop bridge)	TI TMS320C28346 (@300MHz) based self-designed computer
Onboard Graphics Processing Unit	NVIDIA Jetson TX2 with 256 CUDA Cores
Laser Rangefinder	Astech LDS-30A
Camera	Logitech C920 HD webcam

B. Flight Experiment Results

Images in Fig. 12 to Fig. 14 show the flying UAV and transmitted scenes that onboard GPU is processing during flight experiments. Graphs in Fig. 15 to Fig. 17 are flight trajectories based on the GPS measurements. In Fig. 12 to Fig. 14, green box indicates target tracker is activated, and the center of the box is used for calculating the difference dX_I and dY_I. Red arrow starting from the image center marker '+' tells body velocity of the UAV.

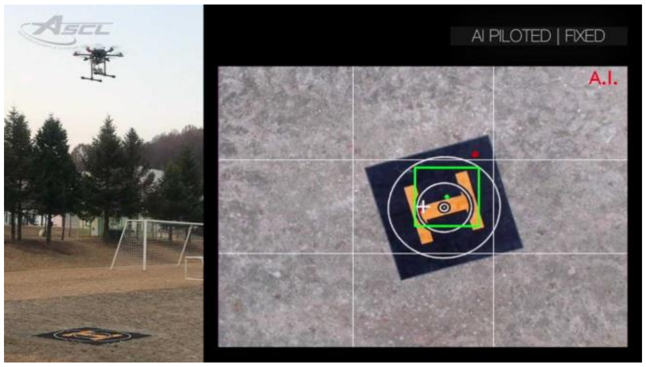

Fig. 12. Scenes taken from flight experiments (fixed target 1).

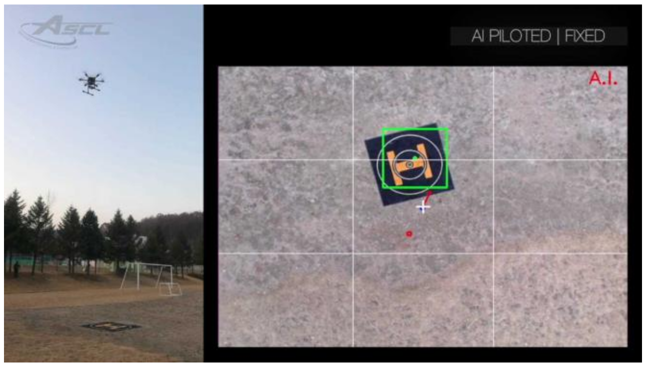

Fig. 13. Scenes taken from flight experiments (fixed target 2).

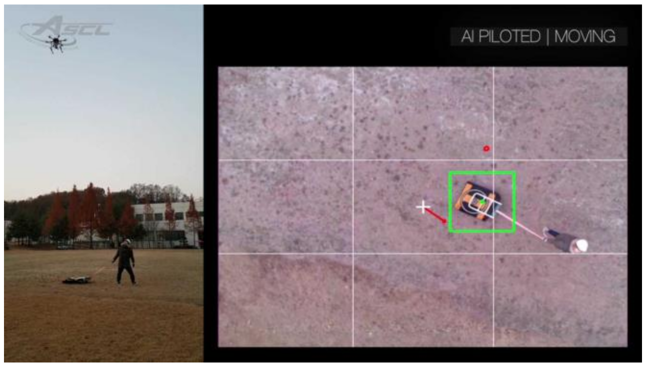

Fig. 14. Scenes taken from flight experiments (moving target).

Based on the aforementioned information along with the altitude and Inertial Navigation System (INS) measurements, the AI suggests desired roll and pitch commands to the attitude controller as much as the red dot deviates from the center marker. Although we manually shut off the AI pilot below the altitude of 1.5m due to safety reasons e.g., final approach attitude and motor control is not studied, we were able to confirm that the AI pilot can substitute human pilots or other human-made guidance algorithms using vision sensor during the landing approach of multi-copter UAV.

Unlike the results in Fig. 8 to Fig. 10, the non-smooth trajectories in Figs. 15 to Fig. 17 can be explained in the following three reasons: The one is the low-performance target tracker during the flight experiments. Unlike the simulated learning environment where the accurate image location of the landing site is given, the actual flight test has oscillating image location by using a simple type of target tracking method with K-Nearest Neighbor (KNN) algorithm. The second reason is that the processing time of an image along with the calculation of actor neural networks, and transmission time in serial communications make up to 200ms delay in total. The last one is the disturbances in the real-world environments. Since the training environment does not model gusts trajectories form the flight experiments could be non-smooth ones.

V. Concluding Remarks

We presented a vision-based autonomous landing of a multi-copter UAV by means of the reinforcement learning. We built a simulation environment and constructed a framework to train and apply our AI pilot on existing UAV control system to introduce a guidance method that performs the same function of human-designed algorithms with less effort. Since we observed promising results both from the simulation and flight experiments, we are going to compare the AI pilot and/or vision-based guidance algorithms built for the same purpose by establishing some performance criteria in the near future.

Acknowledgment

This work has been supported by Korea Advanced Institute of Science and Technology (KAIST) as an artificial intelligence research for the 4th industrial revolution.

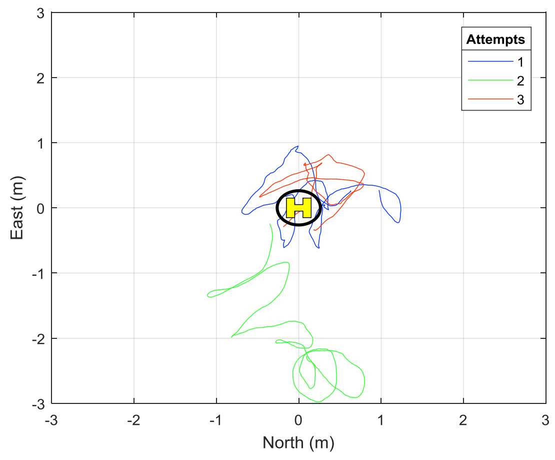

Fig. 15. Flight trajectoreis druing experiments (top view)

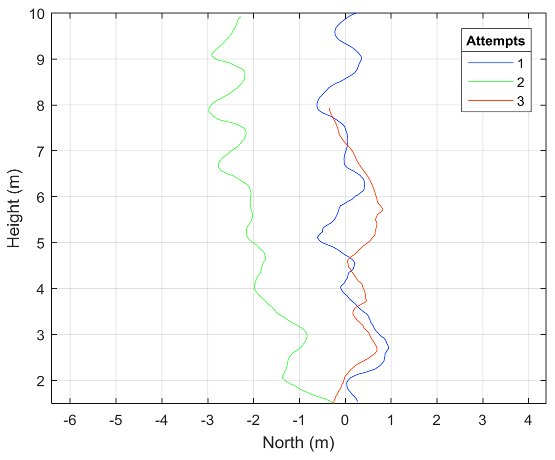

Fig. 16. Flight trajectoreis druing experiments (side view 1).

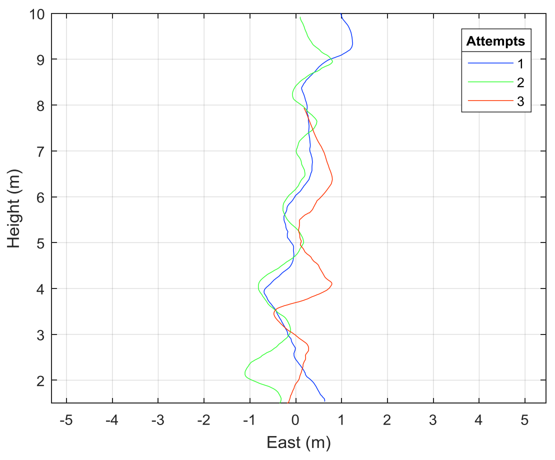

Fig. 17. Flight trajectoreis druing experiments (side view 2).

References

[1] B. Canis, *Unmanned aircraft systems (UAS): Commercial outlook for a new industry*. Congressional Research Service Washington, 2015.

[2] Y. C. Bi and H. B. Duan, "Implementation of autonomous visual tracking and landing for a low-cost quadrotor," (in English), *Optik,* vol. 124, no. 18, pp. 3296-3300, 2013.

[3] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 2009, pp. 1-6: IEEE.

[4] T. Baca, P. Stepan, and M. Saska, "Autonomous landing on a moving car with unmanned aerial vehicle," in *Mobile Robots (ECMR), 2017 European Conference on*, 2017, pp. 1-6: IEEE.

[5] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually guided landing of an unmanned aerial vehicle," *IEEE transactions on robotics and automation,* vol. 19, no. 3, pp. 371-380, 2003.

[6] X. Yang, H. Pota, M. Garratt, and V. Ugrinovskii, "Prediction of vertical motions for landing operations of uavs," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2008, pp. 5048-5053: IEEE.

[7] B. Herisse, T. Hamel, R. Mahony, and F. X. Russotto, "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow," (in English), *Ieee Transactions on Robotics,* vol. 28, no. 1, pp. 77-89, Feb 2012.

[8] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine,* vol. 17, no. 3, pp. 56-65, 2010.

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

[10] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, p. 529, 2015.

[11] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, no. 7587, pp. 484-489, 2016.

[12] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.