33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

SAFE REINFORCEMENT LEARNING BASED MULTI-ROTOR COLLISION AVOIDANCE WITH UNEXPECTED OBSTACLES

Hyungjoo Ahn[1], Chulsoo Lim[1], Jayden Dongwoo Lee[1] & Hyochoong Bang[1]

[1]KAIST, Korea Advanced Institute of Science and Technology

Abstract

From warfare to the civil domain, UAV's roles and missions are kept increasing and more complicated. To complete the sophisticated mission, the significance of autonomous flight under uncertain condition rise. For an avoidance problem with the unexpected situation caused by dynamic obstacles on a quad-rotor, methods based on deep reinforcement learning have recently come into the spotlight, even though safety issues to be implemented in the real world are included. In this research, a collision avoidance algorithm is proposed based on reinforcement learning combined with a safety filter, with the advantages of (1)reshaping the unsafe command from the DDPG actor network to avoid collision, and (2)using simplified depthmap for DDPG agent training to consider a real-world implementation and to reduce the computation time. Through the surrounding obstacle detection sensor such as 360° 3D Lidar, the UAV flight control unit can estimate the distance between the airborne vehicle and detect stumbling blocks by depthmap and perform evasion flight if required. For path following considering obstacles, we adopt the Reinforcement Learning (RL) agent. Meanwhile, the RL agent shows excellent accuracy and fast learning compared to optimization-based avoidance it cannot guarantee successful evasion. To ensure safeness and fast learning, we suggest an RL agent with a safety filter. As a result, the RL agent acquired about an 80% success rate on a random-dynamic obstacle mission with randomly generated 1,000 trials. Also, some of the successful scenarios are included to show the agent can avoid the collision with the randomly moving dynamic obstacles with suggested safe reinforcement learning method. The paper is organized as follows; first, we described the problem to solve and clarified the definition of random-dynamic obstacle and environment including quad-rotor dynamic equations. Then, we described a safety filter to reshape the velocity command from reinforcement learning agent not to collide if any collision with obstacles or walls is predicted. Also, Deep Deterministic Policy Gradient (DDPG) neural network based reinforcement learning method with the Actor-Critic Framework for given quad-rotor state and the depthmap is defined for quad-rotor obstacle avoidance. Finally, simulation results based on MATLAB environment, and concluding remarks are given.

Keywords: safe reinforcement learning, collision avoidance, indoor flight, safety filter

1. Introduction

In recent years, small sized Unmanned Aerial Vehicles(UAVs), or Micro Aerial Vehicles(MAVs) are widely used for the various indoor mission including autonomous flight. Compared with the outdoor flight with high altitude, indoor flight is surrounded with the obstacles with higher density. Therefore, obstacle avoidance is necessary for the autonomous flight in indoor environment[2, 16].

There are various obstacle avoidance methods in obstacle dense environment. The main stream can be described with two groups, optimization-based method[3, 5, 17, 22] and machine learning-based method[7, 8, 12]. For the optimization-based obstacle avoidance, it is usually accurate and can guarantee the collision avoidance with the obstacles, but need more computation time compared with machine learning-based method[10]. On the other hand, machine learning-based method is faster than optimization-based method, but is impossible to guarantee the collision avoidance performance.

Meanwhile, the obstacles can be categorized as two groups, known and unknown objects. For the usual situation, known obstacles are given with the mapping algorithm such as Simultaneous Localization And Mapping(SLAM)[4, 9, 11, 13]. As these obstacles are predictable, there is enough computation time to avoid collision because it is able to maneuver more in advance. Also, as the obstacle is provided with a static map, it can be considered as the non-changing constraints. In this case, applying optimization-based method is effective to guarantee the collision avoidance performance.

But for the unknown obstacles, they normally detected with the sensor on the platform when the obstacles get closer to it. Moreover, there also can be moving obstacles which is hard to predict its movement. If an unexpected obstacle is detected, multi-rotor should react immediately to avoid detected obstacles, which implies optimization-based collision avoidance is inappropriate in this situation. Instead, machine learning based obstacle avoidance method can be applied for the faster reaction with unexpected obstacles. However as described above, machine learning-based method is impossible to guarantee success on the collision avoidance. To overcome this issue, safety filter with optimal problem form[6, 19, 20] is recently introduced to reshape the unsafe action created with the Reinforcement Learning Agent (RL-agent).

In this paper, reinforcement learning-based moving obstacle avoidance method with safety filter, or safe-RL, is introduced with a quad-rotor platform in MATLAB environment[1]. The main contributions of this paper are 1) introducing the safety filter to guarantee the non-colliding flight with reinforcement learning based obstacle avoidance, and 2) using simplified depthmap instead of original depthmap to minimize both the gap between reality and simulation and the size of the network to reduce the calculation time.

2. Problem Definition

As assuming the indoor flight, flyable area limited with the room $\mathbf{S}_{room} \subset \mathbb{R}^3$ is defined in an inertial coordinate limited within a square area as below,

$$\mathbf{S}_{room} = [P_{min}, P_{max}] \tag{1}$$

where

$$P_{min} = [X_{min}, Y_{min}, Z_{min}],\ P_{max} = [X_{max}, Y_{max}, Z_{max}]. \tag{2}$$

With the assumption that the walking people to avoid are simplified as cylinders, N_{obst} number of moving obstacles are existing inside $\mathbf{S}_{room}$. The position, radius and the height of ${n_{obst,i}}^{th}$ obstacle at time t_c is defined as $\mathbf{P}_{obst,i}(t_c) = [X_{obst,i}(t_c), Y_{obst,i}(t_c), 0]$, $R_{obst,i}$ and $H_{obst,i} = Z_{max} - Z_{min}$, respectively. Also, a union space of the all obstacles at time t_c is expressed as $\mathbf{S}_{obst}(t_c) \subset \mathbf{S}_{room}$.

Then, safe zone that the quad-rotor can fly at time t_c is defined with a safe zone $\mathbf{S}_{safe}(t_c)$ as below.

$$\mathbf{S}_{safe}(t_c) = \mathbf{S}_{room} - \mathbf{S}_{obst}(t_c) \tag{3}$$

Current position of a quad-rotor at time t_c in inertial frame $\mathbf{p}(t_c)$ is given as appeared in the follows.

$$\mathbf{p}(t_c) = [X(t_c), Y(t_c), Z(t_c)] \in \mathbf{S}_{safe} \tag{4}$$

Also, let velocity in inertial frame and attitude as follows.

$$\mathbf{v}(t_c) = [v_x(t_c),\ v_y(t_c),\ v_z(t_c)] \tag{5}$$

$$\Theta(t_c) = [\phi(t_c),\ \theta(t_c),\ \psi(t_c)] \tag{6}$$

Then, $\mathbf{v}(t_c)$ is limited with the performance specification of quad-rotor as follows.

$$\mathbf{v}(t_c) \in [\mathbf{v}_{min},\ \mathbf{v}_{max}] \tag{7}$$

where

$$\mathbf{v}_{min} = [v_{x,min}, v_{y,min}, v_{z,min}], \ \mathbf{v}_{max} = [v_{x,max}, v_{y,max}, v_{z,max}]. \tag{8}$$

The discretized dynamics and controller of the UAV can be described as

$$\mathbf{x}(t_c + 1) = f(\mathbf{x}(t_c), \mathbf{U}(t_c)) \tag{9}$$

where $\mathbf{x}(t_c)$ and $\mathbf{U}(t_c)$ are the states and the reference command with current time t_c. For $\mathbf{x}(t_c)$, position in inertial frame, velocity, Euler angles, and angular rates are included. The observation space that can given to the reinforcement learning framework can be given as $\mathbf{S}_t$. $\mathbf{A}(t_c)$ is an appropriate velocity command that can be applied to the platform to avoid obstacle and reach to the global goal position $\mathbf{p}_{goal}$. The diagram of defined problem is as follows.

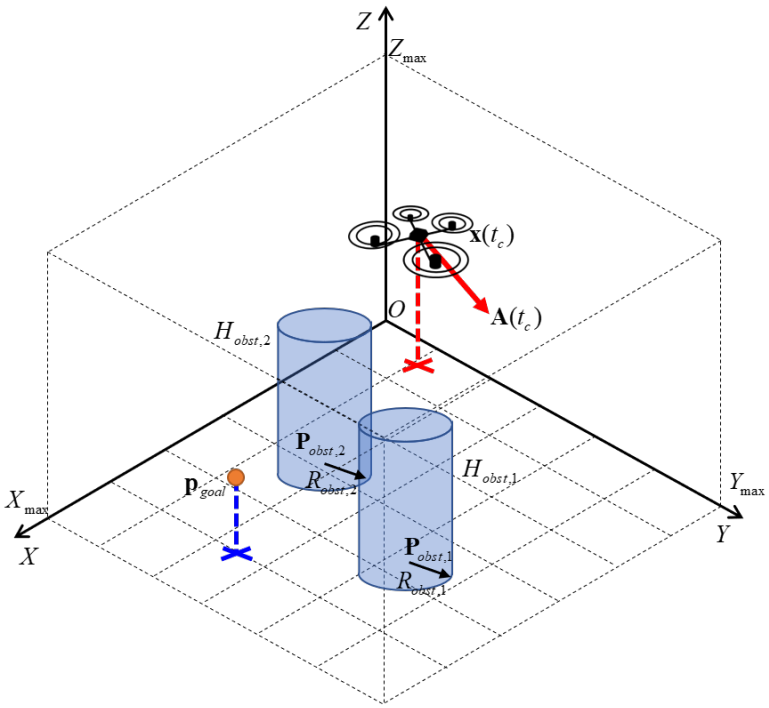

Figure 1 – Diagram of defined problem.

3. Environment Settings

3.1 Workspace and Obstacles

The definition of obstacles at time t_c and the workspace are given in chapter 2. For the discretized obstacle movement simulation, acceleration is changing with the randomized magnitude $a_{mag,i}(t_c) = a_{mag,i,max}u_{rand}$ and the azimuth angle, $a_{azi,i}(t_c) = a_{azi,i}(t_c - 1) + a_{azi,i,max}u_{rand}$ where $a_{mag,i,max}$, $a_{azi,i,max}$, and u_{rand} are the maximum magnitude and azimuth angle of acceleration vector, and a random variable which follows the uniform distribution with $u_{rand} \in [-0.5, 0.5]$, respectively. With the randomly generated $a_{mag,i}(t_c)$ and $a_{azi,i}(t_c)$, the velocity and the position of the obstacle is propagated with the assumption of constant velocity and acceleration in the current time with the discretized time step.
To limit the velocity of each obstacles as the maximum obstacle speed $v_{obst,max}$, the speed is revised to $v_{obst,rev}$ which is smaller than $v_{obst,max}$, and the direction is changed to opposite. Also, if the obstacles are crashed with each other, the direction of the velocity are changed to the other side of each others which is parallel to the line connecting two obstacles in the crashing moment.

3.2 Depthmap with Obstacles

The depthmap is simulated with the assumption of 360-degree camera. Distance with the obstacles are calculated by checking the distance between the quad-rotor and cross point on surface of the obstacles with the simulated ray on each pixel[18]. With the pixel number of $N_{hor} \times N_{ver} = 20 \times 20$, and

the Field Of View(FOV) of $\alpha = 2\pi$ and $\beta = \pi/6$ in the horizontal and vertical direction of the camera, we can get a depthmap in t_c as $I_D(t_c, \alpha, \beta, N_{hor}, N_{ver})$.

3.3 Multi-rotor Dynamics and Controller

The Equation-Of-Motion(EOM) for the quad-rotor can be described with translational and rotational motion[14]. The EOM in translational and rotational motion is defined as below

$$m\ddot{\mathbf{p}}(t_c) = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ T_{sum} \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & -c_\phi s_\theta \\ 0 & 1 & s_\phi \\ s_\theta & 0 & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{11}$$

$$\mathbf{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathbf{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad \mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{12}$$

where m, g, I_{kk}, T_{sum}, p, q, r and M_k are the mass of quad-rotor, gravitational acceleration, moment of inertia in k axis, total thrust from motors, angular rates in roll, pitch, and yaw directions, and moment in k axis, respectively. Finally, controller is designed with cascade form with PID(Proportional-Integral-Derivative) controllers. With the given velocity command in X and Y axis, P-controller generates the roll and pitch angle command. Yaw angle command is generated with an outer algorithm. Then, roll, pitch, and yaw angle command are given as an input for a PID-controller to generate angular rate command which are given as the input of a PID-controller to generate moment in (X-Y-Z) axis. On the other hand, velocity command in Z axis is generated from a P-controller with a given altitude command. Then, the total thrust is calculated from a P-controller with the velocity command in Z axis. Finally, Total the state of next step is calculated with the dynamics in simulator with given total thrust and the moment from the controllers.

From the RL-agent, the calibrating value for $v_{x,des}$ and $v_{y,des}$, $\Delta v_{x,des}$ and $\Delta v_{y,des}$ are generated, which is provided to the controller. Then, $U(t_c)$ can be defined as below.

$$\mathbf{U}(t_c) = \begin{bmatrix} v_{x,des} + \Delta v_{x,des} \\ v_{y,des} + \Delta v_{y,des} \\ Z_{des} \\ \psi_{des} \end{bmatrix}. \tag{13}$$

In this paper, action at current time, $\mathbf{A}(t_c) = [\Delta v_{X,des}, \Delta v_{Y,des}]^T$, is generated with RL-agent. With the definition of relative position from current quad-rotor position to the global goal position as $\mathbf{p}_{rel}(t_c) = [X_{rel}(t_c), Y_{rel}(t_c), Z_{rel}(t_c)]^T = \mathbf{p}(t_c) - \mathbf{p}_{goal}$, rest of the reference commands are generated based on algorithm as $[v_{x,des}, v_{y,des}]^T = [v_{x,forward}, 0]^T$, $\psi_{des} = atan2(Y_{rel}(t_c), X_{rel}(t_c))$ to make quad-rotor can get closer to the final goal without $\mathbf{A}(t_c)$. For the velocity or commands such as $v_x(t_c)$, $v_y(t_c)$, $v_z(t_c)$, $v_{x,des}(t_c)$, $v_{y,des}(t_c)$, $\Delta v_{x,des}(t_c)$, and $\Delta v_{y,des}(t_c)$, they use $(x_B - y_B - z_B)$ coordinate system as shown in the figure, that $(x_B - y_B)$ plane is parallel with $(X - Y)$ plane with rotated angle $\psi(t_c)$, and z_B axis is equals to Z axis. In this paper, equation of motion and the controller with given action $\mathbf{A}_t(t_c)$ in discrete world is represented as a discretized function in Eq. 9.

4. Safety Filter

4.1 Definition of Safety Filter

Safety filter is defined to reshape the generated action into safe region if any unexpected unsafe situation is predicted such as collision[6, 19, 20]. To predict the collision, next position at time $t_c + 1$, $\tilde{\mathbf{p}}(t_c + 1)$ is predicted based on the nominal equation of motion from Eqs. 10-12 and the controller described in chapter 3.3 with given action $\mathbf{A}_t(t_c)$. The distance with the nearest obstacle from predicted quad-rotor position $\tilde{\mathbf{p}}(t_c + 1)$ is expressed in the below

$$d_{obst,near}(t_c) = \|\tilde{\mathbf{p}}(t_c+1) - \mathbf{p}_{obst,near}(t_c)\|_2 - R_{obst,near} \tag{14}$$

where $\mathbf{p}_{obst,near}(t_c)$ and $R_{obst,near}$ are the position and radius of the nearest obstacle from the quad-rotor, respectively. Also, the nearest distance in (X-Y) plane between four walls and $\tilde{\mathbf{p}}(t_c+1)$ is defined as $d_{wall,near}(t_c)$. If the crash with wall or obstacle is detected, action is reshaped with an optimal action $\mathbf{A}_t^*$ to avoid collision. The safety filter algorithm is described as follows.

$$\mathbf{A}_t^*(t_c) = \begin{cases} safetyFilter(\mathbf{S}_t(t_c), \mathbf{A}_t(t_c)), & \text{if } min(d_{obst,near}(t_c), d_{wall,near}(t_c)) < 0 \\ \mathbf{A}_t(t_c), & \text{else} \end{cases} \tag{15}$$

To find an optimal value $\mathbf{A}_t^*(t_c)$, $safetyFilter(\mathbf{S}_t(t_c), \mathbf{A}_t(t_c))$ is defined with a given problem in the below.

$$\begin{gathered} \underset{\mathbf{S}_t(t_c), \mathbf{A}_t(t_c)}{\text{Minimize}} \quad \|\mathbf{A}_t(t_c)) - \mathbf{A}_t^*(t_c))\|_2^2 \\ \text{subject to} \\ \mathbf{x}(t_c+1) = f(\mathbf{x}(t_c), \mathbf{U}(t_c)) \\ \mathbf{p}(t_c+1) \in \mathbf{S}_{safe}, \;\; \mathbf{v}(t_c+1) \in [\mathbf{v}_{min}, \; \mathbf{v}_{max}] \end{gathered} \tag{16}$$

5. Reinforcement Learning

For many problems related to multi-rotor autonomous flight or collision avoidance for mobile robots, Reinforcement Learning (RL) is used such as vision based autonomous landing[14] or obstacle avoidance for ground mobile robot[7, 8, 12]. As similar to the problems introduced in the above, RL is also used to the problem in this paper to make the additional velocity command $\mathbf{A}(t_c)$ for collision avoidance of multi-rotor. Like other RL problems, RL-agent with environment is introduced to make action with given observations and reward such as state of the quad-rotor or the depthmap of surrounding obstacles, and to simulate or calculate the next state and the reward of the quad-rotor with given action. For the deep RL framework, Deep Deterministic Policy Gradient(DDPG) is introduced with Actor-Critic network[14, 15] based on MATLAB RL toolbox, example with DDPG agent training for swing up and balance pendulum[1].

5.1 Network Structure

The structure of actor network is divided into two paths, image path and attitude path. For image path, one Convolutional Layer(CL), and one Fully Connected Layer(FCL) are included with one Rectified Linear activation Unit(ReLU) function for activation. Also for the attitude path, one FCL is included. Two paths are concatenated and connected with three FCLs with hidden layer sizes of 400, 300, and the dimension of action for each. Two ReLU functions are used for the activation and one hyper tangent function is used to reshape the output size as between -1 and 1. Finally, scaling function added in the end to scale the output size as the maximum action sizes.
The critic network is divided into three paths, image path, attitude path, and action path. Image path and attitude path are the same with the actor network. Action path is constructed with a FCL, size of 300. Three path are concatenated and ReLU function with FCL are used for the critic output. Actor and critic layers are shown in the Fig. 2-3.
Adaptive Moment estimation (Adam) optimizer[21] is used with learning rate 10^{-4} and 10^{-3} for action and critic networks, experience buffer length with 10^6, discount factor with 0.99, mini batch size of 256, and noise variance and decay rate with 0.6 and 10^{-6}. The maximum step number in one episode $t_{step,max}$ is defined that the episode is finished when the agent didn't reach the terminal condition within $t_{step,max}$.

5.2 Actor-Critic Framework

For the deep-RL framework with DDPG, actor and critic network is needed. For the training process, actor network create an action $\mathbf{A}_t(t_c)$ with given observation $\mathbf{S}_t(t_c)$. Then, critic network receives both $\mathbf{A}_t(t_c)$ and $\mathbf{S}_t(t_c)$ to update the policy of actor. After the training process is finished, only actor is activated to generate appropriate $\mathbf{A}_t(t_c)$ with given $\mathbf{S}_t(t_c)$ while critic is deactivated.

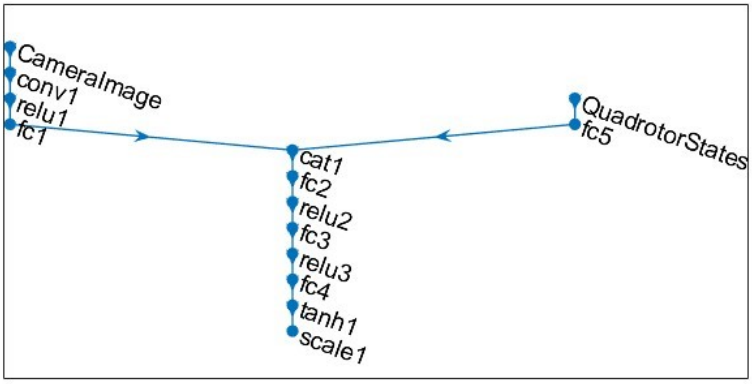

Figure 2 – Structure of actor network.

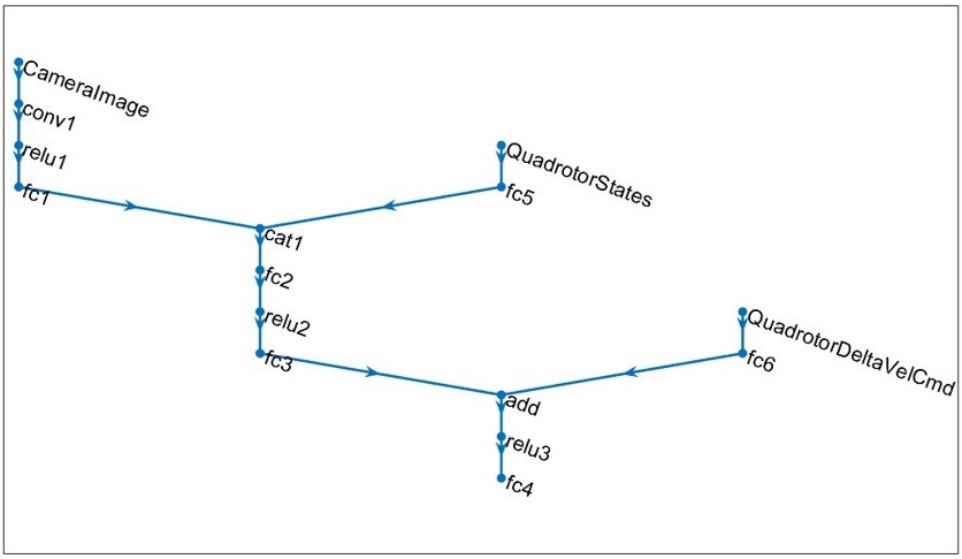

Figure 3 – Structure of critic network.

The observation is defined from the quad-rotor state and the depthmap. For the quad-rotor state in observation, position in (X-Y) plane, relative position with goal position in (X-Y) plane, heading angle, and velocity in (X-Y) plane, and a depthmap with normalization are included as follows.

$$\mathbf{S}_t(t_c) = [I_D(t_c),\ X_N(t_c),\ Y_N(t_c),\ X_{N,rel}(t_c),\ Y_{N,rel}(t_c),\ \psi_N(t_c),\ v_{N,x}(t_c),\ v_{N,y}(t_c)] \tag{17}$$

As prescribed in the above, action $\mathbf{A}_t(t_c)$ creating with the actor network is defined as the additional velocity in $(x_B - y_B)$ coordinate, $\Delta v_{X,des}$ and $\Delta v_{Y,des}$, to avoid collision and reach the goal position.
To check the successability and end the episode while training, '$IsDone$' variable is defined with a few scenarios as follows.

$$IsDone = (IsReachedGoal)\ |\ (IsCrashedWall)\ |\ (IsCrashedObstacle) \tag{18}$$

Reward $R_t(t_c)$ is defined as a summation of few sub-rewards as the below

$$R_t(t_c) = R_{obst}(t_c) + R_{goal}(t_c) + R_{wall}(t_c) + R_{obstDist}(t_c) + R_{action}(t_c) + R_{state}(t_c) + R_{time}(t_c) \tag{19}$$

$$R_{obst}(t_c) = -10^6 \times (IsCrashedObstacle) \tag{20}$$

$$R_{goal}(t_c) = \begin{cases} 2, & \text{if } \|[X_{N,rel}(t_c),\ Y_{N,rel}(t_c)]\|_2^2 < \varepsilon_{near} \\ 10^6, & \text{elseif } IsReachedGoal \end{cases} \tag{21}$$

$$R_{wall}(t_c) = \begin{cases} -50, & \text{if } min(X_N(t_c), Y_N(t_c), (X_{max}-X_{min})-X_N(t_c),\ (Y_{max}-Y_{min})-Y_N(t_c)) < \varepsilon_{near} \\ -10^6, & \text{elseif } IsCrashedWall \end{cases} \tag{22}$$

$$R_{obstDist}(t_c) = -\left\{ \frac{d_{max}/2 - \|\mathbf{p}(t_c) - \mathbf{p}_{obst,near}(t_c)\|_2 - R_{obst,near}}{2.5} \right\}^2 - d_{max} \tag{23}$$

$$R_{action}(t_c) = -\left\| \left[\frac{10\Delta v_{x,des}}{(v_{max}-v_{min})},\ \frac{3\Delta v_{y,des}}{2(v_{max}-v_{min})} \right] \right\|_2^2 \tag{24}$$

$$R_{state}(t_c) = 25\left(1 - 0.5\,\|[X_{N,rel}(t_c),\ Y_{N,rel}(t_c)]\|_2^2\right) \tag{25}$$

$$R_{time}(t_c) = \begin{cases} -2\times 10^6 \left(1 - \dfrac{t_{step,fin}}{t_{step,max}}\right), & \text{if } (IsCrashedWall) \mid (IsCrashedObstacle) \\ 2\times 10^6 \left(1 - \dfrac{t_{step,fin}}{t_{step,max}}\right), & \text{elseif } (IsReachedGoal) \\ 0, & \text{else} \end{cases} \tag{26}$$

where $\varepsilon_{near} = 0.05$, $t_{step,max} = 3000$, and $t_{step,fin}$ is the time step number until the episode finished.

5.3 Reinforcement Learning Framework

With the described environment, safety filter, and RL-agent, reinforcement learning framework are interacting for an RL framework. As the actor from 5. create an action $\mathbf{A}_t(t_c)$, safety filter in chapter 4. reshape the given $\mathbf{A}_t(t_c)$ as $\mathbf{A}_t^*(t_c)$ if unsafe condition is detected. Next observation and reward with quad-rotor state and depthmap is generated with the given environment in chapter 3. During training, safety filter is excluded for agent to learn the proper action value in randomized environment. And after training, safety filter is included to prevent the unsafe situation while critic network is excluded. Also, actor network is deactivated if I_D has only the maximum depth value that there is no need to avoid obstacles. The overall framework with RL-agent and safety filter, environment is shown in the figure below[20].

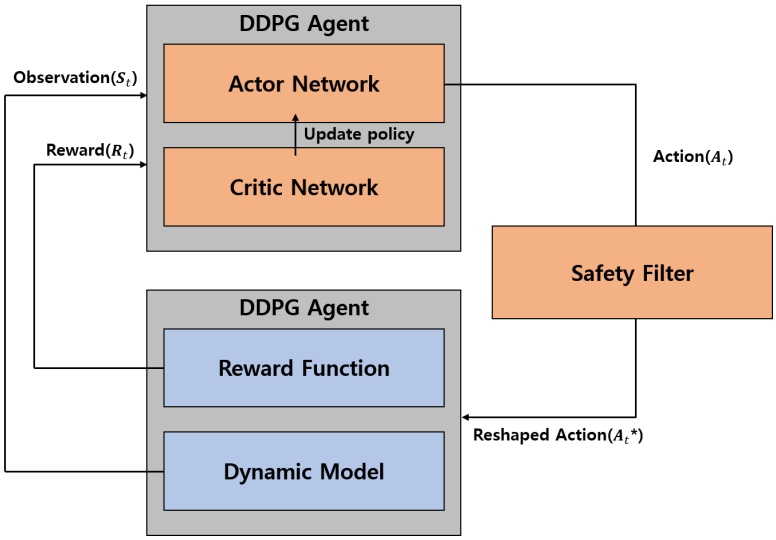

Figure 4 – Network structure of actor and critic network.

6. Simulation Results

With the defined problem in the above, autonomous flight simulation to reach the given global goal position including safe-RL based obstacle avoidance is done with MATLAB 2020a with Intel i7-8700K CPU. GPU is not used for training as the DDPG agent network is not big enough to The training is done with 11,000 episodes to learn environment for agent, and safety filter is applied with the trained actor from DDPG agent.

For the verification of the performance of safety-RL framework, simulation is done with the randomized environment with obstacle number and start position, room size, quad-rotor start and global goal positions. The result of randomized 1000 trials is shown as the table below.

Table 1 – Simulation result with 1000 trials in randomized scenarios

Cases	Succeed	Crashed with obstacle	Crashed with wall	Total
Number	790	136	74	1000

Also, the succeeded result including trajectory of quad-rotor and the obstacles, intermediate steps of obstacle avoidance, and the final state of the scenario is appeared in each figures in the below. The trajectories of quad-rotor and the obstacles, start and final goal positions of quad-rotor, current positions of quad-rotor are obstacles are marked as red and black solid lines blue 'o' and red 'x' markers, black and blue 'x' markers, respectively.

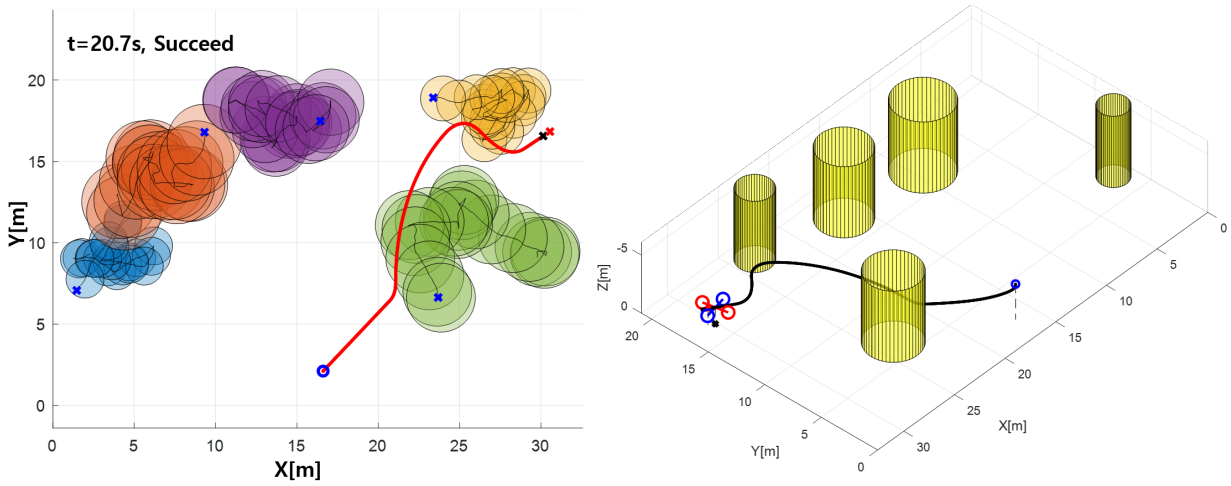

Figure 5 – Succeeded scenario 1, Final state with trajectories

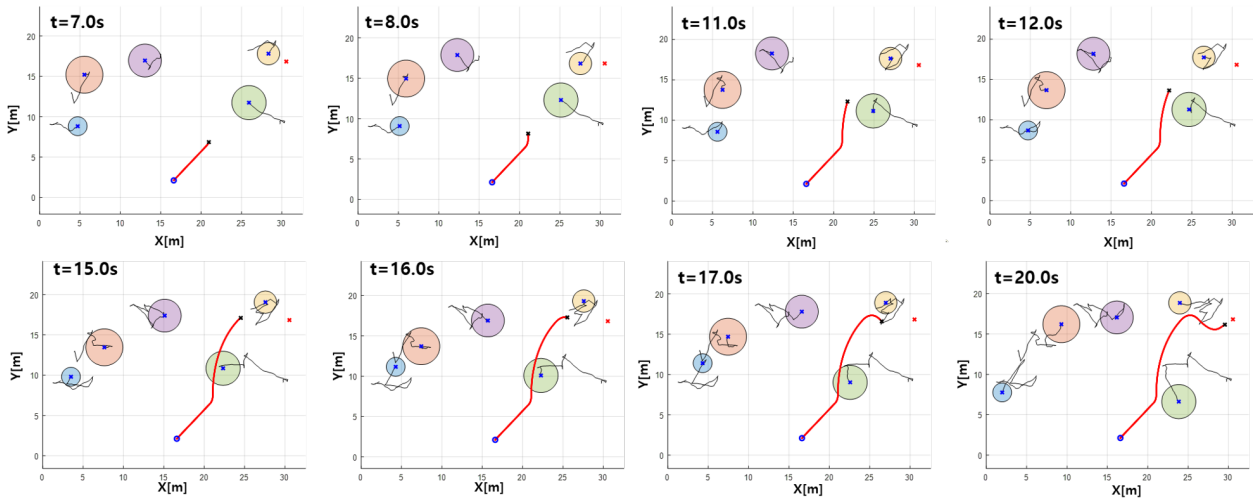

Figure 6 – Succeeded scenario 1, Intermediate states with specific times

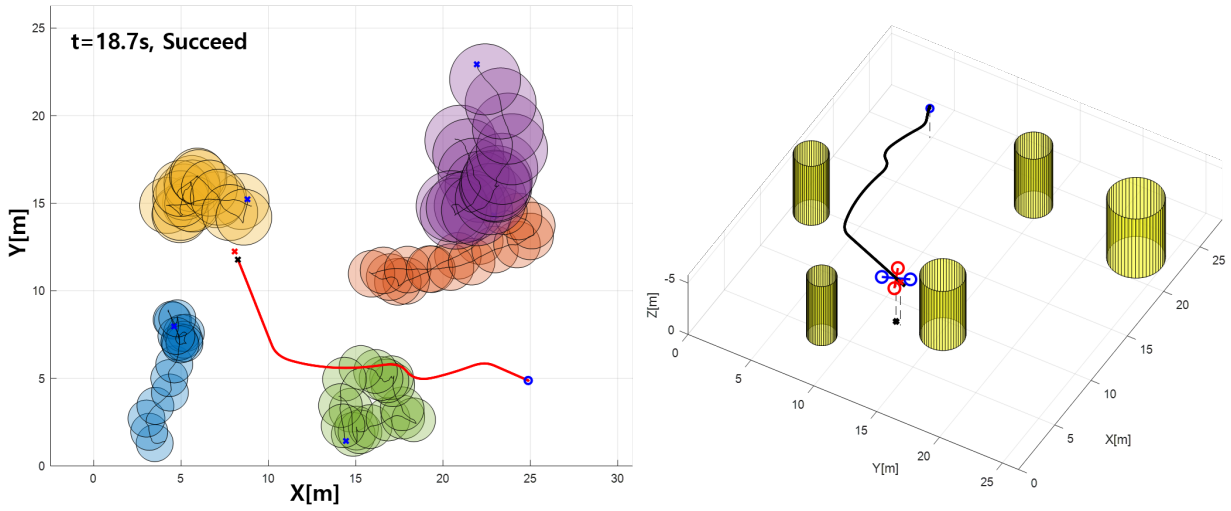

Figure 7 – Succeeded scenario 2, Final state with trajectories

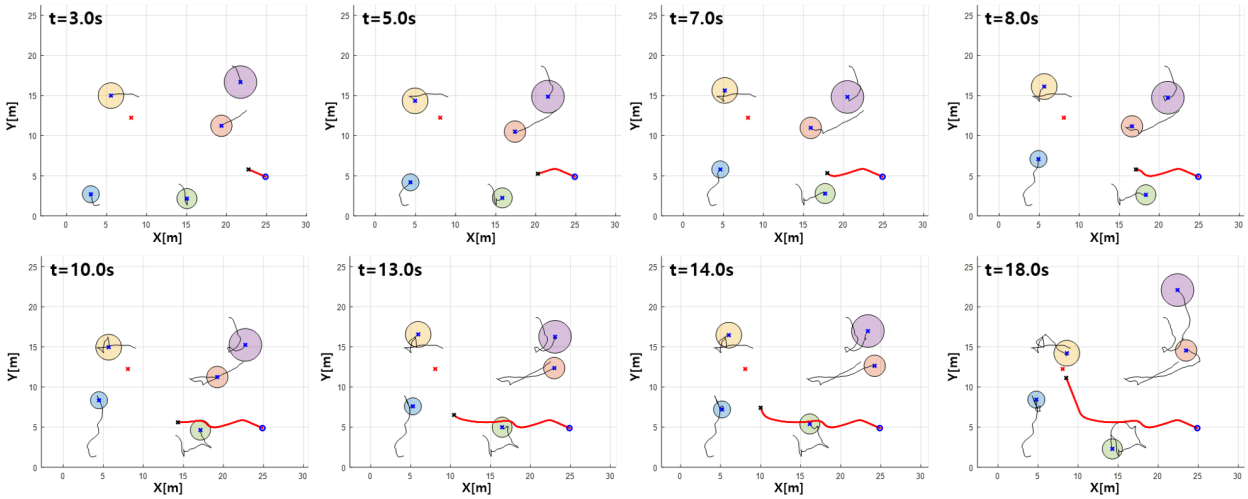

Figure 8 – Succeeded scenario 2, Intermediate states with specific times

As shown in the table 1, avoidance maneuvers to reach the goal position are possible with a high probability. Also, it is possible to see the agent tries to avoid the detected obstacles with depthmap and finally reach the global goal position as appeared in Fig.5-8.

7. Summary and Conclusions

In this paper, obstacle avoidance framework is suggested including RL-agent and safety filter. Environment of the framework is described with the equation of motion of quad-rotor. Safety filter is defined to reshape the action value to prevent the unsafe situation. Then, Deep-RL agent is defined with the actor and critic network with the appropriate observation including normalized position, relative position with global goal, heading angle, and velocity of the quad-rotor. Finally, safe-RL framework is constructed and trained for the actor to create additional velocity command to avoid obstacle while quad-rotor is heading to the global goal position. Safe-RL framework is conducted with MATLAB environment simulator including quad-rotor dynamics. With the simulation result, we can conclude that that quad-rotor can reach the global goal position while avoiding the unexpectedly detected moving obstacles with low computation time. Also, successiveness on the collision avoidance is performed. With the randomized 1,000 situations, the RL agent acquired about an 80% success rate on a random-dynamic obstacle mission by adopting safety filter with RL-agent.

8. Contact Author Email Address

If you have any questions, please send an email to:
'baoro1995@ kaist.ac.kr' (the first author) or 'hcbang@kaist.ac.kr' (corresponding author).

9. Copyright Statement

References

[1] *MATLAB Reinforcement Learning Toolbox™ User's Guide R2022a*, pages 5–60, 5–66. The Mathworks, Inc., Natick, Massachusetts, United States, 2022.

[2] H. J. Ahn, J. W. Park, H. C. Bang, and Y. S. Kim. Model predictive control-based multirotor three-dimensional motion planning with point cloud obstacle. *Journal of Aerospace Information Systems*, 19 (3):179–193, 2022.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon path planning for 3d exploration and surface inspection. *Autonomous Robots*, 42(2):291–306, 2018. doi: 10.1007/s10514-016-9610-0.

[4] A. M. Brandt and M. B. Colton. Haptic collision avoidance for a remotely operated quadrotor uav in indoor environments. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2724–2731. IEEE, 2010. doi: 10.1109/ICSMC.2010.5641798.

[5] Y. Chen, M. Cutler, and J. P. How. Decoupled multiagent path planning via incremental sequential convex programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961. IEEE, 2015. doi: 10.1109/ICRA.2015.7140034.

[6] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 01, pages 3387–3395, 2019.

[7] R. Cimurs, J. H. Lee, and I. H. Suh. Goal-oriented obstacle avoidance with deep reinforcement learning in continuous action space. *Electronics*, 9(3):411, 2020.

[8] M. Duguleana and G. Mogan. Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications*, 62:104–115, 2016.

[9] Y. Endo, K. Sato, A. Yamashita, and K. Matsubayashi. Indoor positioning and obstacle detection for visually impaired navigation system based on lsd-slam. In *2017 International Conference on Biometrics and Kansei Engineering (ICBAKE)*, pages 158–162. IEEE, 2017. doi: 10.1109/ICBAKE.2017.8090635.

[10] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2008.

[11] O. Esrafilian and H. D. Taghirad. Autonomous flight and obstacle avoidance of a quadrotor by monocular slam. In *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, pages 240–245. IEEE, 2016. doi: 10.1109/ICRoM.2016.7886853.

[12] B.-Q. Huang, G.-Y. Cao, and M. Guo. Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In *2005 International conference on machine learning and cybernetics*, volume 1, pages 85–89. IEEE, 2005.

[13] S. Huh, D. H. Shim, and J. Kim. Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of uavs. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3158–3163. IEEE, 2013. doi: 10.1109/IROS.2013.6696805.

[14] S. H. Lee, T. M. Shim, S. J. Kim, J. W. Park, K. W. Hong, and H. C. Bang. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 108–114. IEEE, 2018.

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

[16] S. Liu. *Motion Planning for Micro Aerial Vehicles*. PhD thesis, Publicly Accessible Penn Dissertations, University of Pennsylvania, 2018.

[17] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017. doi: 10.1109/LRA.2017.2663526.

[18] J. Park and Y. Kim. Horizontal-vertical guidance of quadrotor for obstacle shape mapping. *IEEE Transactions on Aerospace and Electronic Systems*, 52(6):3024–3035, 2016.

[19] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

[20] K. P. Wabersich and M. N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

[21] Z. Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. Ieee, 2018.

[22] V. V. Zinage and S. Ghosh. Generalized shape expansion-based motion planning for uavs in three dimensional obstacle-cluttered environment. In *AIAA Scitech 2020 Forum*, page 0860, 2020. doi: 10.2514/6.2020-0860.