Autonomous Navigation of UAV Using Deep Reinforcement Learning

Benchun Zhou

School of Automation Science and Electronic Engineering
Beihang University

CONTENTS

CONTENTS

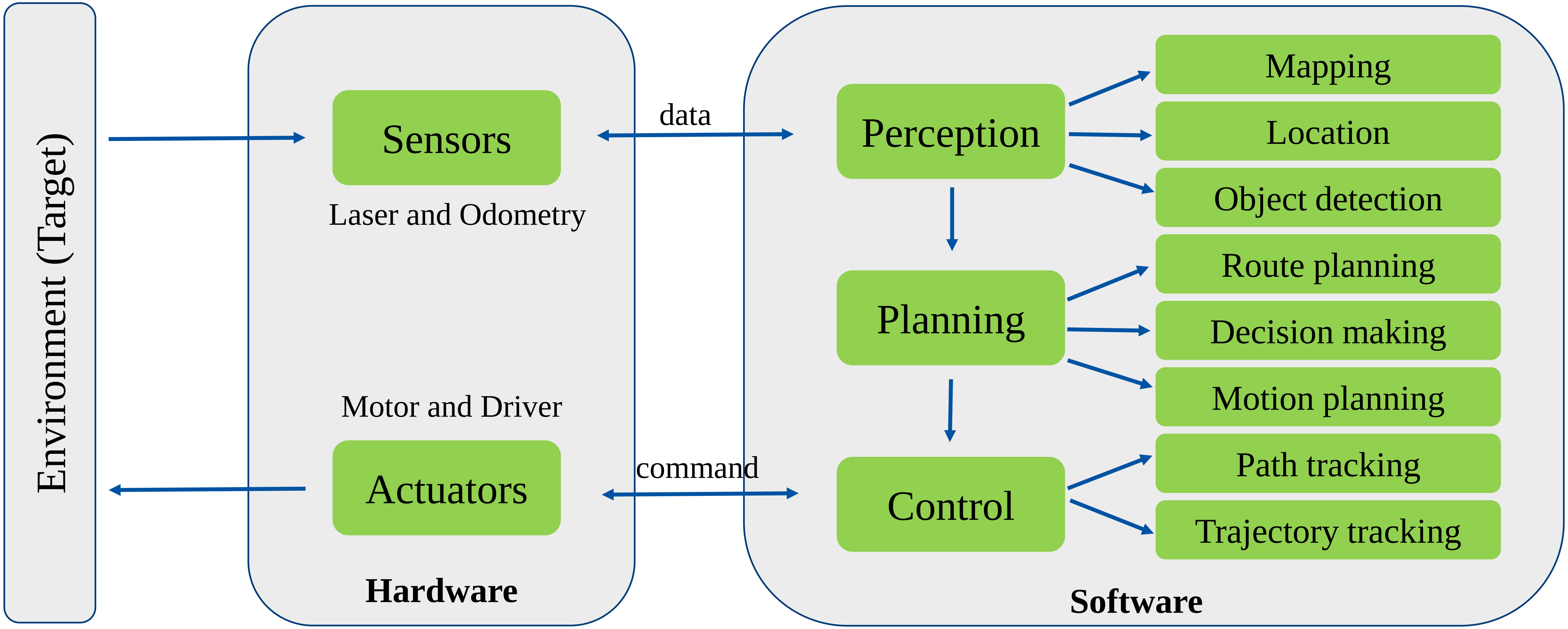
1. Introduction

- **1.1 Autonomous UAV Navigation**
- Application of autonomous unmanned aerial vehicle (UAV)
  - Drone delivery: delivering goods in cities
  - Rescue mission: carrying medical supplies
  - Aerial photography: capturing and recording views

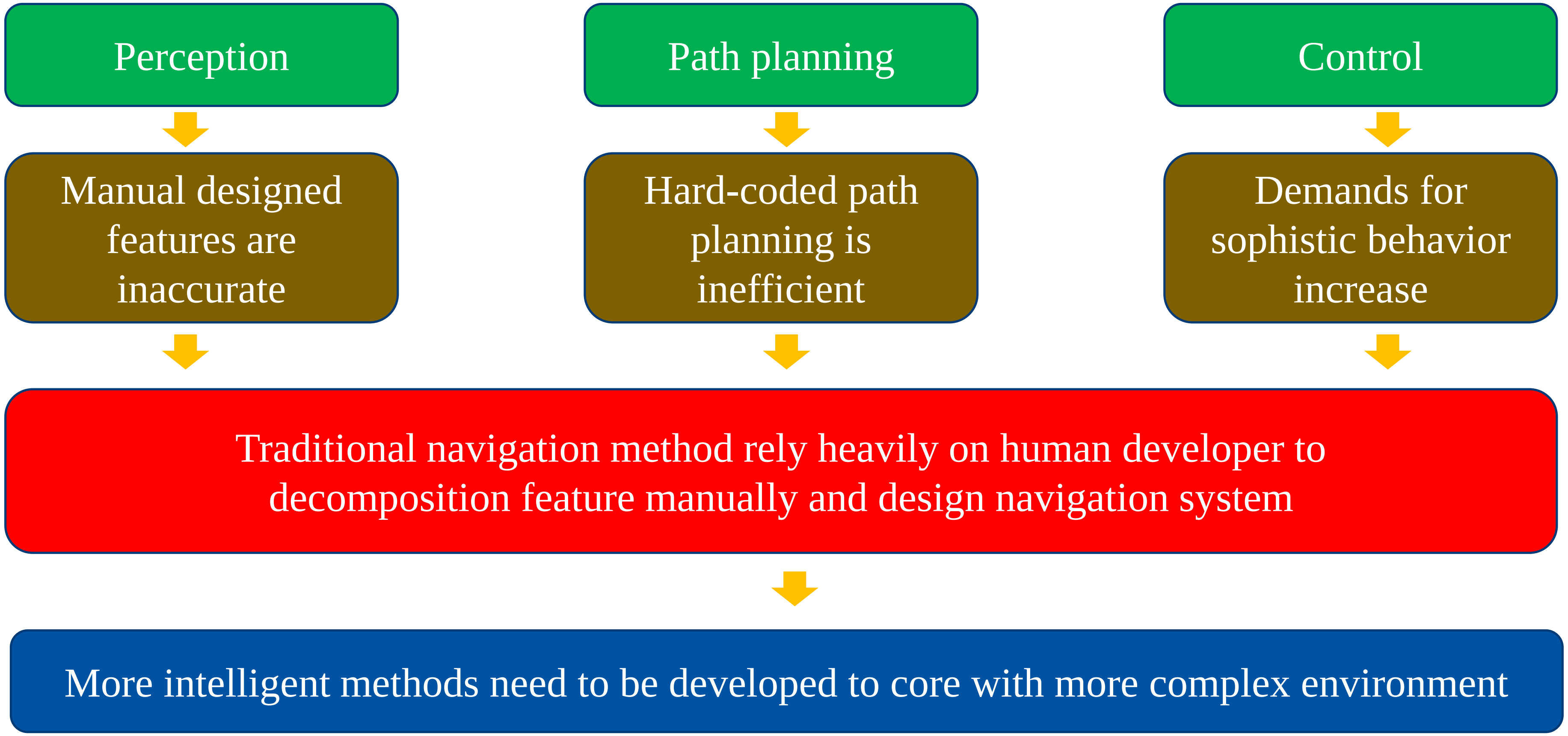
We use drone in the paper

- **1.1 Autonomous UAV Navigation**

- Navigation: a UAV makes a plan on how to safely and quickly reach target location



- **1.1 Autonomous UAV Navigation**
- Challenges



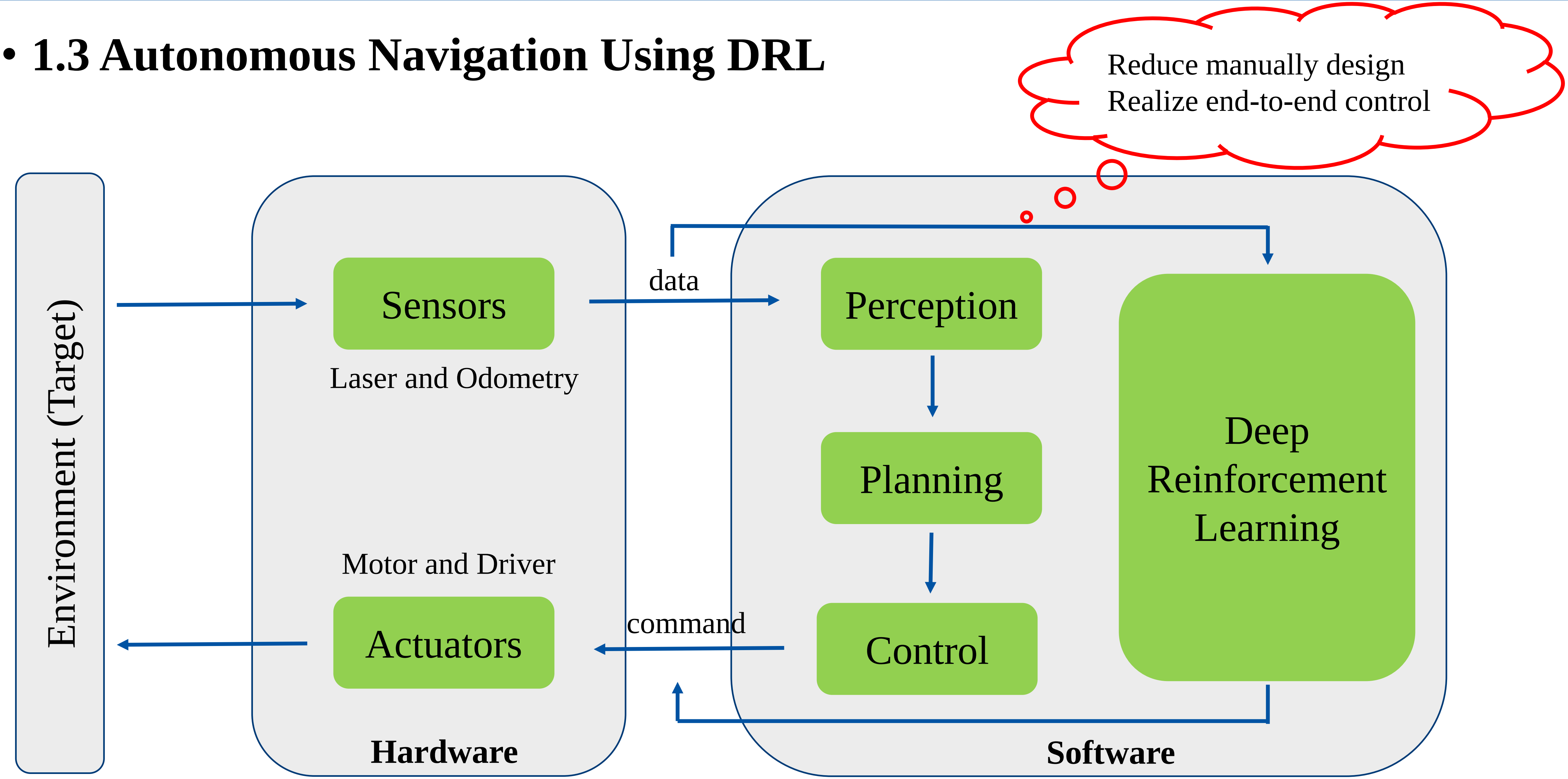
• 1.2 Deep Reinforcement Learning

Reinforcement Learning:
Autonomously learn optimal behavior through trail-and-error interactions with environment.

Applications

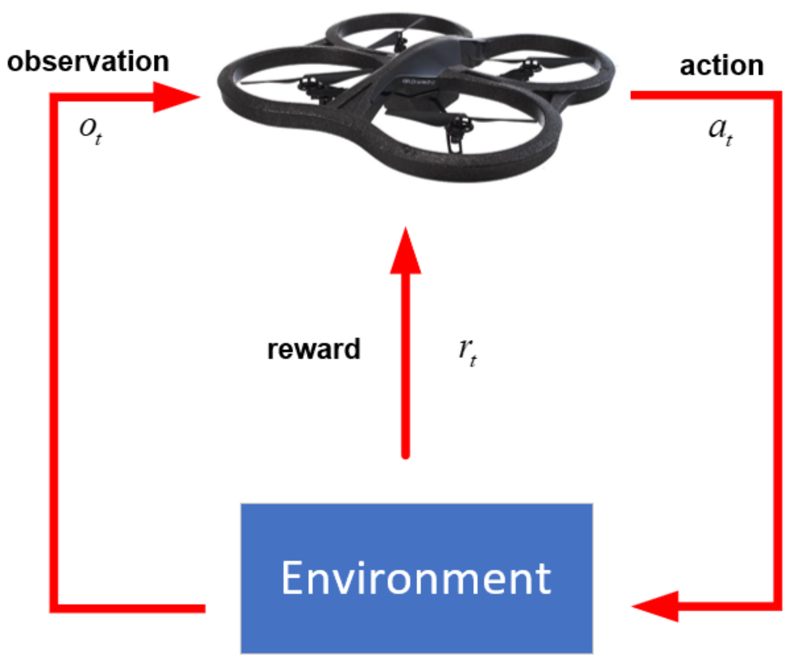
- Play games: Atari, Go, …
- Autonomous driving： Racing Car, ACC, …
- Robot Control: Manipulation, Humanoid, …
- Neural language process: Translator, Data analysis, …

- **1.3 Autonomous Navigation Using DRL**



CONTENTS

• 2.1 Agent and Environment



- The interaction between agent and environment can be describe as:
- At each step t the agent:
  - ➢ Receives an observation o_t
  - ➢ Selects an action a_t following a policy π
  - ➢ Receives scalar reward r_t
  - ➢ Transitions to next state s_{t+1}
- The objective of reinforcement learning is to maximize expected discounted reward through interaction, which was defined as

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

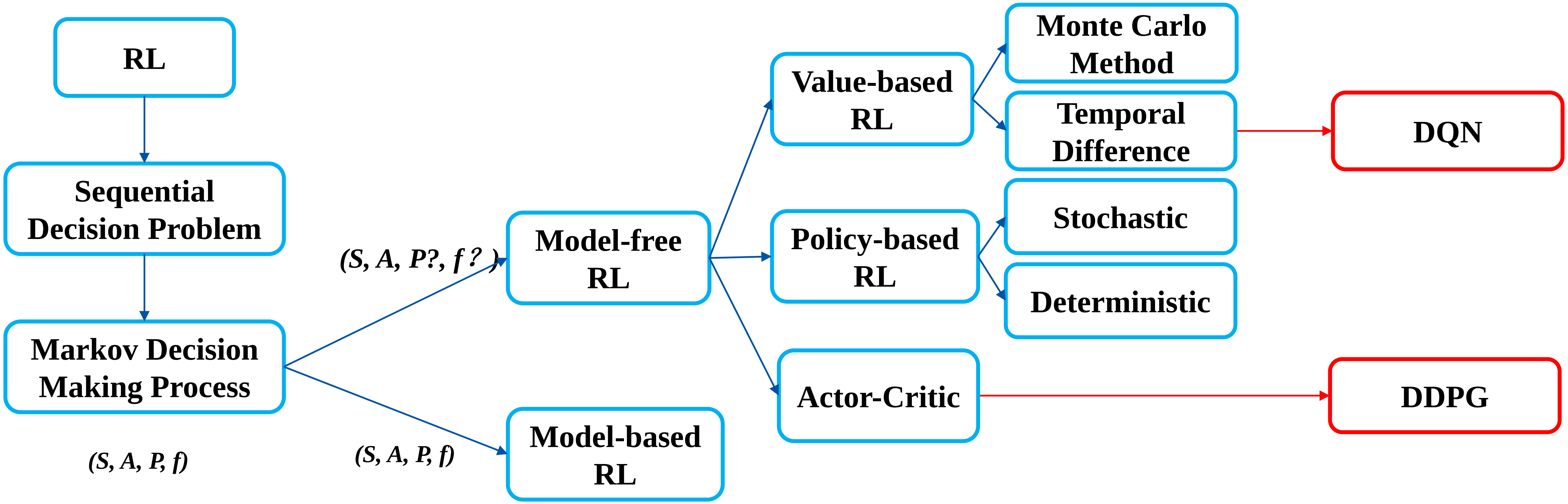
- Policy π is a behaviour function selecting actions given states

$$a = \pi(s)$$

- Action-value function $Q^\pi(s,a)$ is expected future reward by taking the action from a from state s and following policy π:

$$Q^\pi(s,a) = \mathrm{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s,a\right]$$

- **2.2 Reinforcement learning**
  - Reinforcement Learning a general-purpose framework for decision-making



• 2.3 Deep Learning

- Deep Learning is a general-purpose framework for representation learning

Classification

Deep Learning (DL)

- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Deep Residual Network (DRN)
- Generative Adversarial Network (GAN)

Application

- Computer Vision (CS)
- Speech Recognition
- Object Detection & Tracking
- Natural Language Processing (NLP)

• 2.4 Deep Reinforcement Learning

Reinforcement Learning (decision-making) + Deep Learning (perception) = Deep Reinforcement Learning (general intelligence)

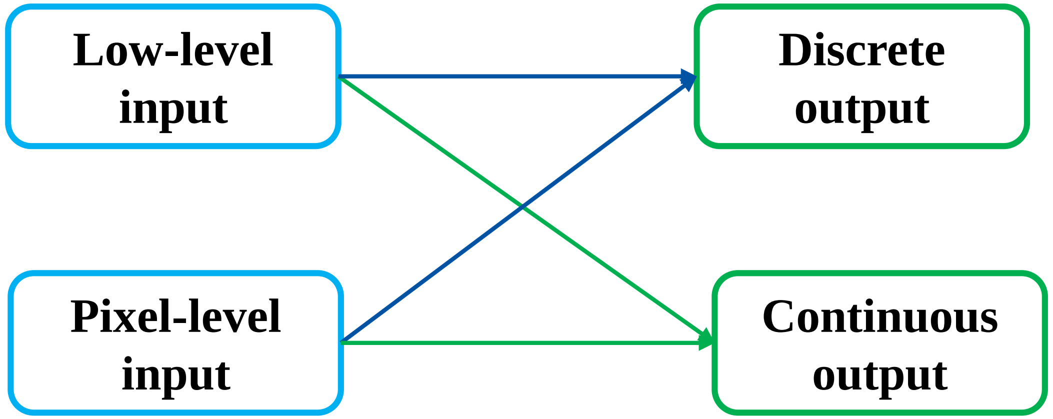
Value-based

- Deep Q-Network (DQN)
- Double Deep Q-Network (DDQN)
- Deep Recurrent Q-Network (DRQN)
- Dueling Deep Q-Network（Dueling DQN）

Policy-based

- Deep Deterministic Policy Gradient (DDPG)
- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)
- Guided Policy Search (GPS)

Control angle



CONTENTS

• 3.1 Value-based Reinforcement Learning

- Based on estimating the values of being a give state, then extracting the policy from the estimated values
  - Estimate the optimal value function $Q^*(s,a)$

$$Q^*(s,a) = \mathrm{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a \right]$$

  - Update the value function by temporal difference error

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(\underbrace{r_t + \gamma \max_{a} Q(s',a')}_{target} - Q(s,a) \right)$$

  - Table Q-learning

	A1	A2	A3	……
S1	*Q(S1,A1)*	*Q(S1,A2)*	*Q(S1,A3)*	
S2	*Q(S2,A1)*	*Q(S2,A2)*	*Q(S2,A3)*	
S3	*Q(S3,A1)*	*Q(S3,A2)*	*Q(S3,A3)*	
……				

State discretization
Action discretion

• 3.2 Deep Q-Network (DQN)

- Deal with state discretization by introducing deep neural network in value function
- Represent value function by deep Q-network with weights w

$$Q(s,a,w) \approx Q^*(s,a) = \mathrm{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a',w) \mid s,a \right]$$

- Define objective function by mean-squared error between Q-target and Q-network

$$L(w) = \mathrm{E}\left[\left(\underbrace{r + \gamma \max_{a} Q(s',a',w)}_{target} - Q(s,a,w) \right)^2 \right]$$

- Optimise objective end-to-end by stochastic gradient descent (SGD)

$$\frac{\partial L(w)}{\partial w} = \mathrm{E}\left[\left(r + \gamma \max_{a} Q(s',a',w) - Q(s,a,w) \right) \frac{\partial Q(s,a,w)}{\partial w} \right]$$

• 3.2 Deep Q-Network (DQN)

- DQN provides a stable solution to deep value-based RL

  1) Use deep neural network with parameters

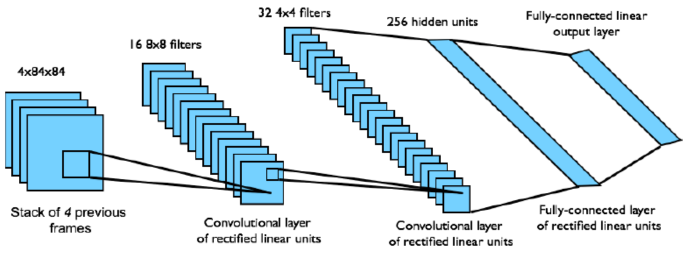
  - Represent value function with deep neural network
  - Deal with pixel-level info

  2) Freeze target Q-network

  - Break correlations between Q-network and target
  - Periodically update Q target using parameter w

  3) Use experience replay

  - Break correlations in data
  - Learn from all past policies



$$r + \gamma \max_{a} Q\left(s', a', w^{-}\right) \leftarrow w$$

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\dots$
$s_t, a_t, r_{t+1}, s_{t+1}$

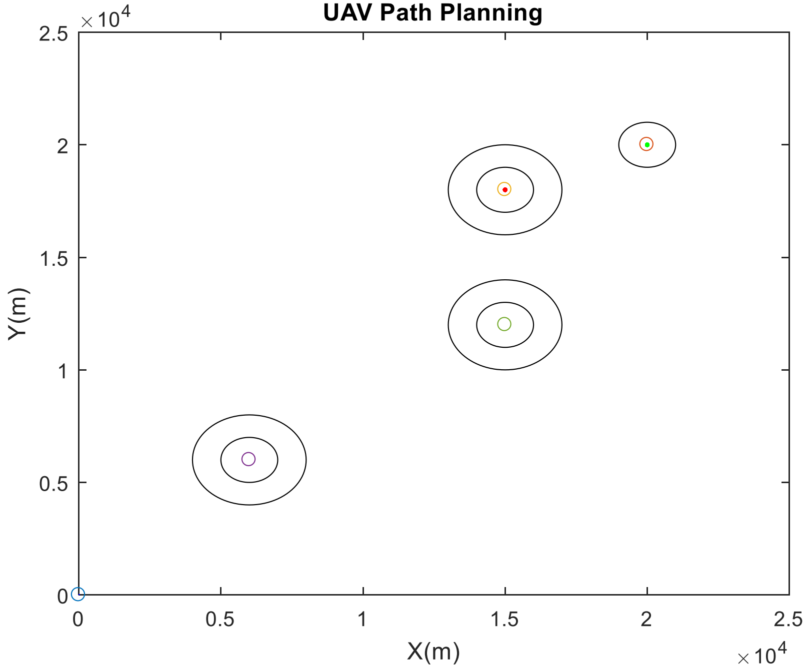
$\rightarrow$ s, a, r, s'

$s_t, a_t, r_{t+1}, s_{t+1}$ $\rightarrow$

Mnih, V, et al. "Human-level Control Through Deep Reinforcement Learning. " *Nature* 518.7540(2015):529.

• 3.3 Planar Navigation Using DQN

➢ 1) Environment



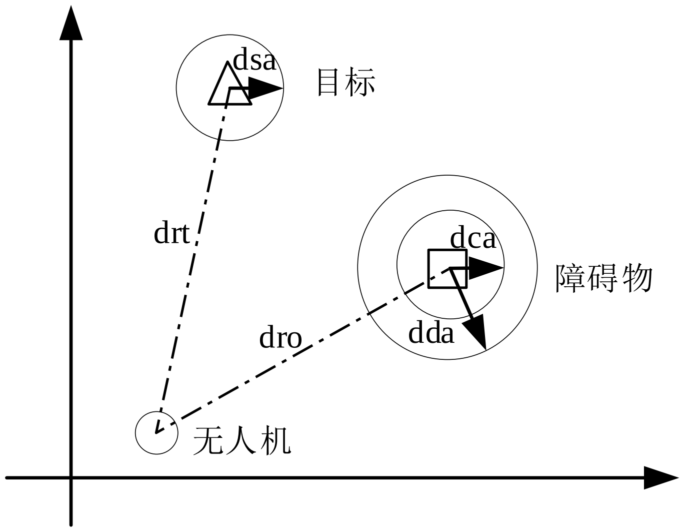
Regarding the UAV as a mass point, employ reinforcement learning method to enable UAV to find a free-collision path toward target.

- ◆ Navigation task：（0km，0km）>>>>>>>>（20km，20km）
- ◆ Environment info: $v_{obstacle} = 200m/s$ $v_{\text{target}} = 200m/s$

  $r_{\det ection} = 1000m$
- ◆ UAV info: $\psi_{init} = 90^{\circ}$ $v_{UAV} = 500m/s$

• 3.3 Planar Navigation Using DQN

➢ 2) Tabular Q learning



□ State discretization

$$S=\begin{cases} WS & d_{rt}\le d_{sa} \\ SS & d_{ro}>d_{da} \\ DS & d_{ca}\le d_{ro}\le d_{da} \\ FS & d_{ro}\le d_{ca} \end{cases}$$

Winning State
Safe State
Dangerous State
Failure State

□ Action discretization {-45°， 0°， +45°}

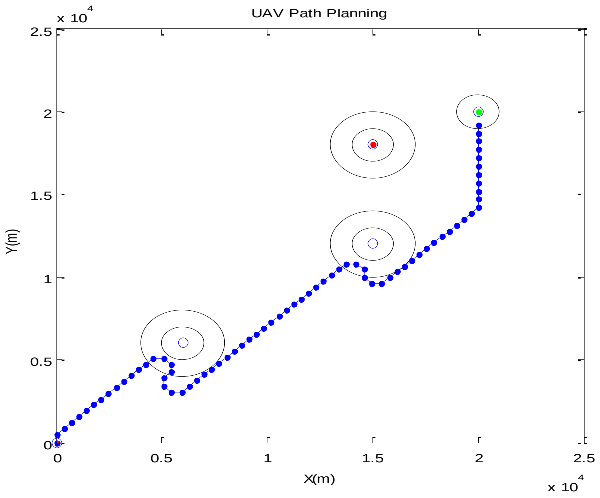
□ Reward discretization

$$r=\begin{cases} 2, & S\subset SS\to WS \\ 1, & S\subset DS\to SS \\ -1, & S\subset SS\to DS \\ -1, & S\subset DS\to DS, d_{ro}(n+1)<d_{ro}(n) \\ 0, & S\subset DS\to DS, d_{ro}(n+1)<d_{ro}(n) \\ -2, & S\subset DS\to FS \end{cases}$$

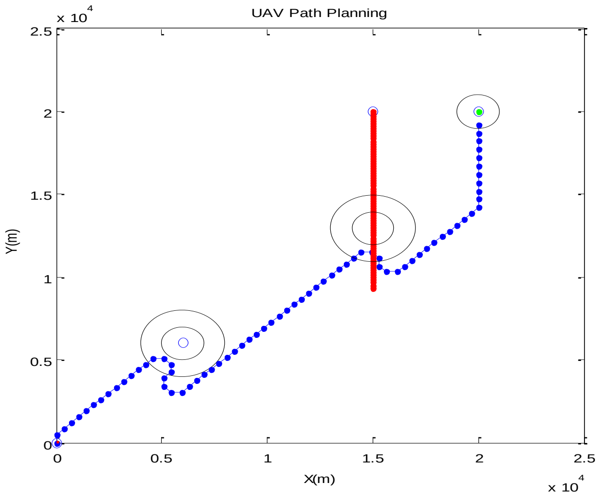
State/Action	Left	Forward	Right
SS	0.2	0.5	0.3
DS	0.7	0.1	0.2
WS	0.3	0.6	0.4
FS	0.7	0.2	0.1

• 3.3 Planar Navigation Using DQN

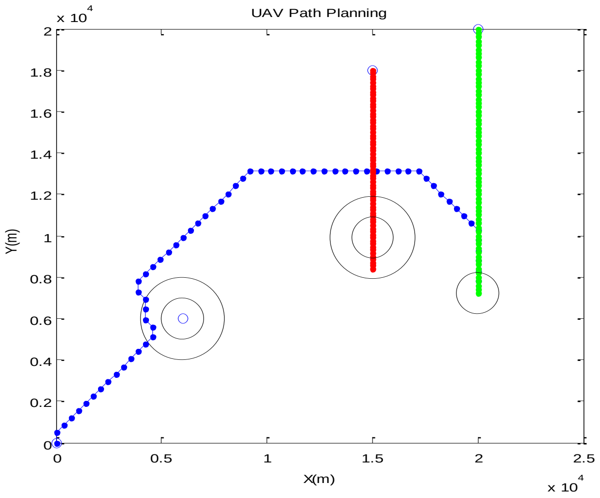
➢ 3) Result



Static obstacle



Moving obstacle



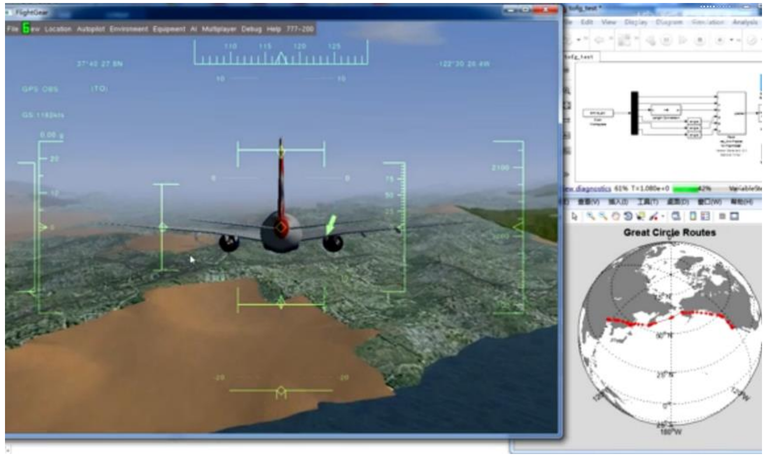
Moving target

- **3.4 Vision-based Navigation Using DQN**

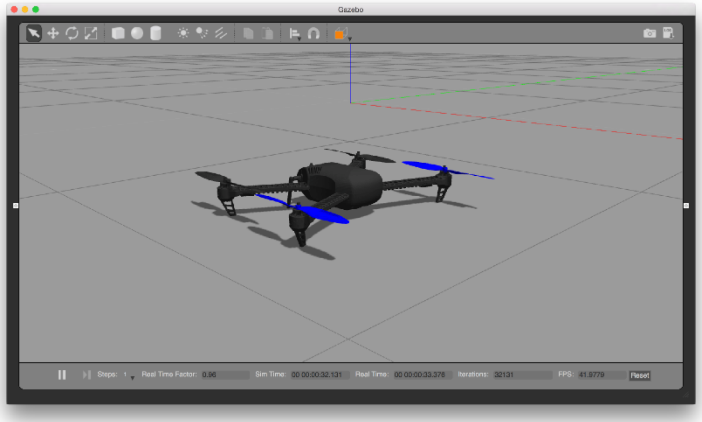
  - 1) Environment

    - Simulator requirements:

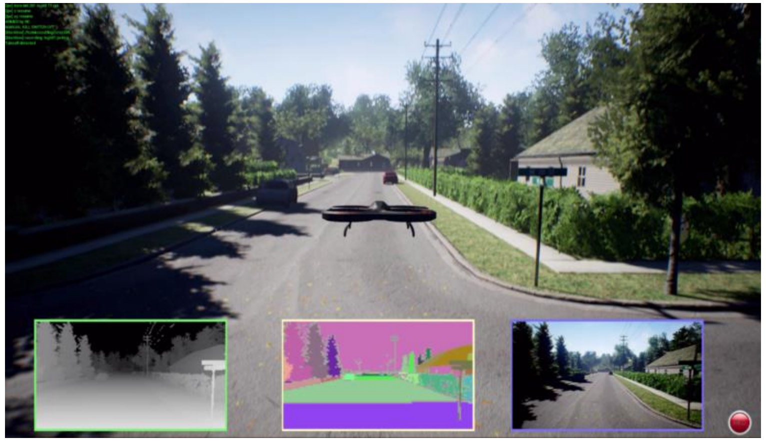
      1. Physical realistic simulation with minimal model errors
      2. Controllable and modifiable environment
      3. Interface to environment (e.g., receive command and send data)



Flight Gear



Gazebo



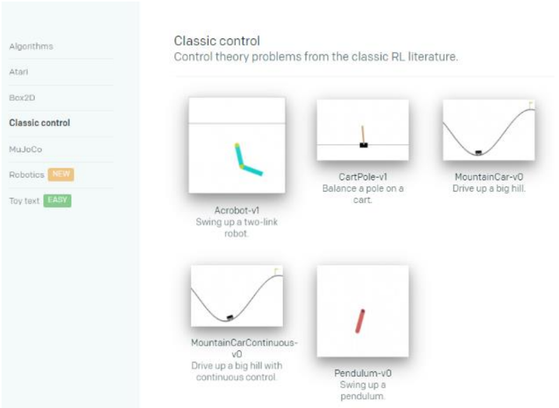
AirSim

• 3.4 Vision-based Navigation Using DQN

➢ 1) Environment

AirSim

OpenAI Gym



Air Gym

Render the environment
Provide API to send data and receive commands

Model the navigation task
Train the agent using reinforcement learning algorithm

• 3.4 Vision-based Navigation Using DQN

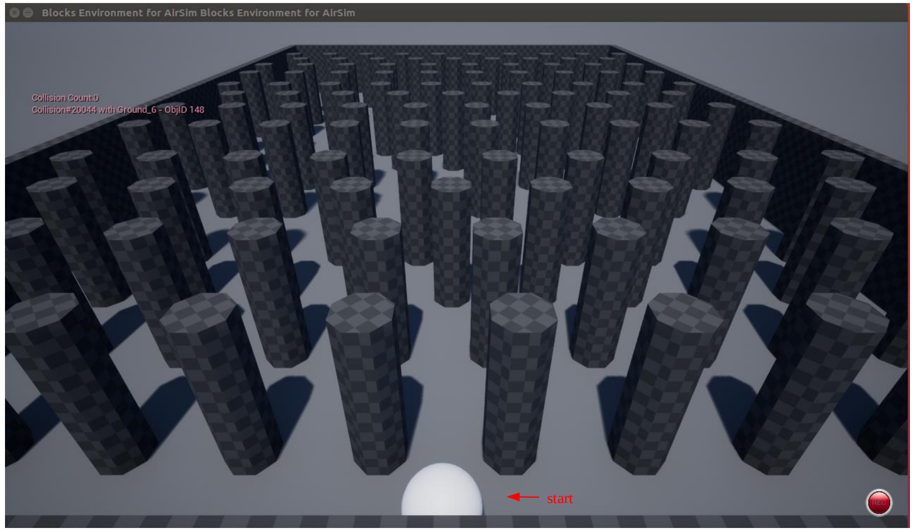
➢ 1) Environment

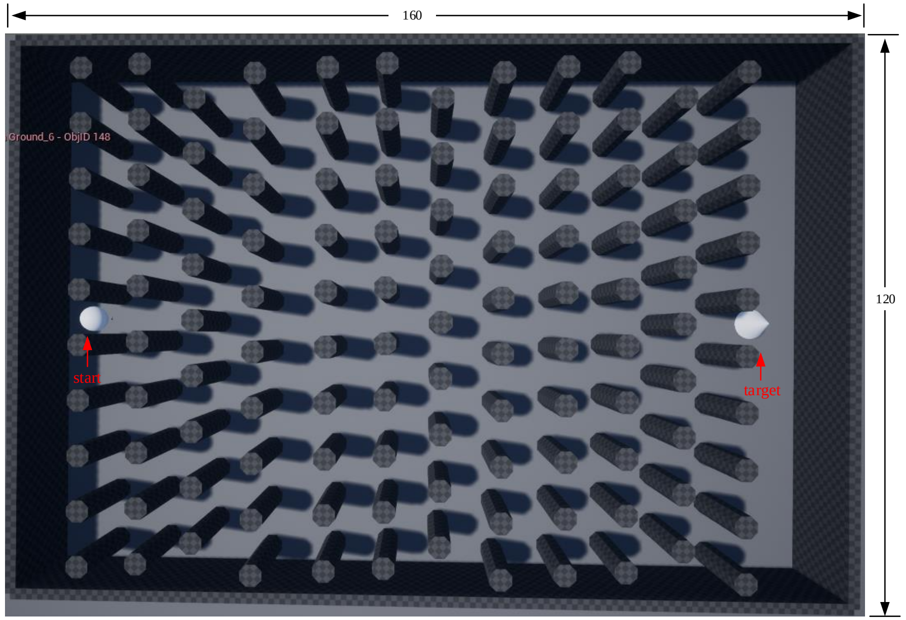
- Navigation Task: UAV has to reach the goal in minimum amount of time without colliding with any obstacle.

(10m, 0m) >>>>>> (150m, 0m)

- Environment information:

Data	Meaning
p_x	Agents global x position
p_y	Agents global y position
p_z	Agents global z position
ψ	Yaw angle relation to initial orientation
Depth Image	Depth image in camera (256×144)
Collided	Boolean collision info





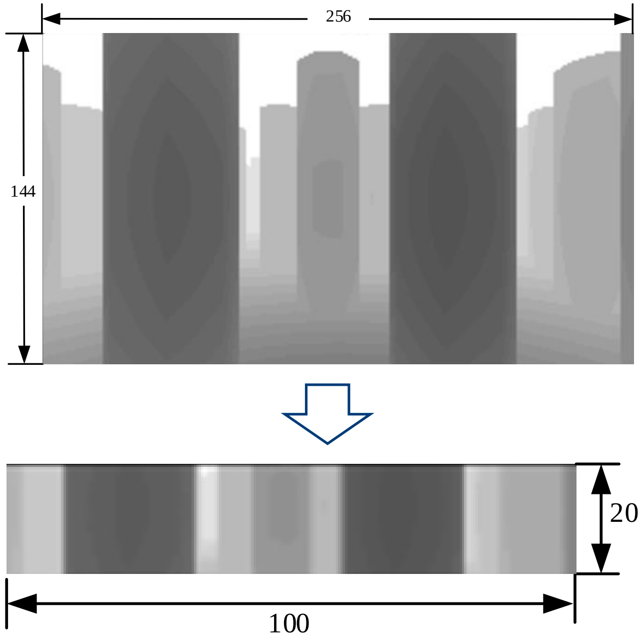
160m*120m*15m

- **3.4 Vision-based Navigation Using DQN**
  - 2) Partial Observation Markov Decision Making Process

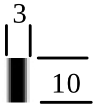
Depth image processed

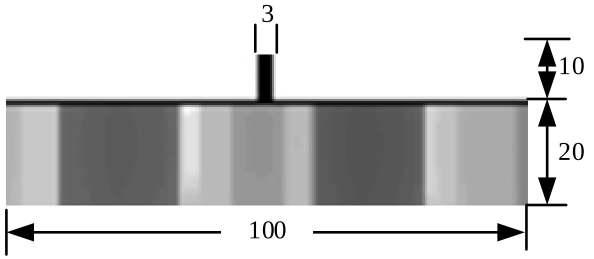
Goal information encoded

State representation



$$\phi = \arctan\left(\frac{g_x - p_x}{g_y - p_y}\right) - \psi$$





• 3.4 Vision-based Navigation Using DQN

➢ 2) Partial Observation Markov Decision Making Process

- Action Space Discretization
  - ➢ UAV flies at fixed level (6m) and at constant speed (4m/s) $a_t \in \{-1, 0, +1\}$
  - ➢ 1) go straight: Move in direction of current heading with 4m/s for 1 s
  - ➢ 2) yaw left: Rotate left with 30° /s for 1 s~=30°
  - ➢ 3) yaw right: Rotate right with 30° /s for 1 s~=30°

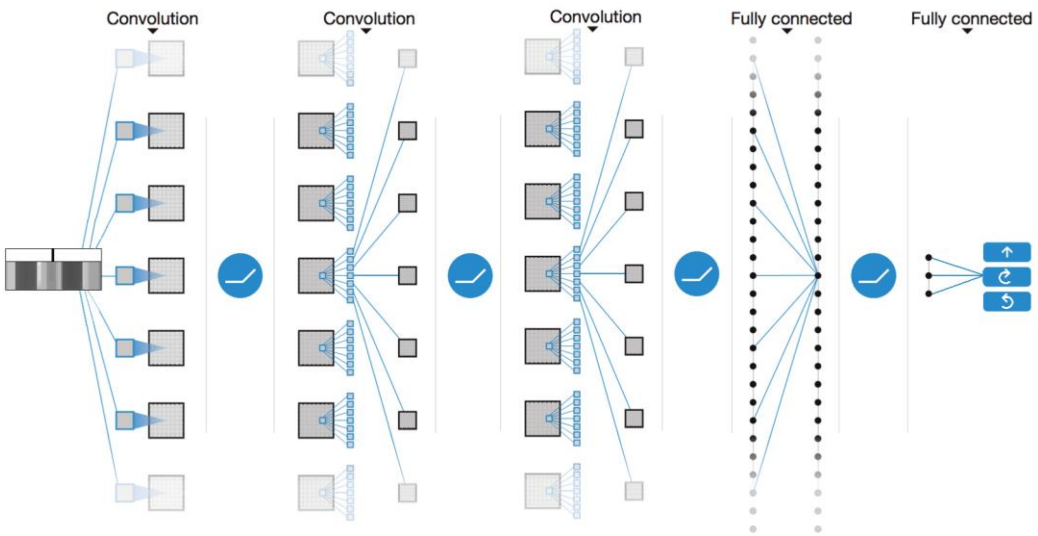
- Reward Function
  - ➢ Consist of terminal reward, time reward, approach reward and track angle reward.

$$r = \begin{cases} 100 & if \quad success == TRUE \\ -100 & if \quad collided == TRUE \\ -1 + \Delta d_{t-1} - \Delta d_t - |\phi| & otherwise \end{cases}$$

• 3.4 Vision-based Navigation Using DQN

➢ 3) DQN Network Architecture

- Architecture
  - The network architecture of DQN consists of 3 convolutional and 2 fully-connect layers
  - Input state is a depth image with 30*100 pixels
  - Output is Q(s,a) for 3 different actions



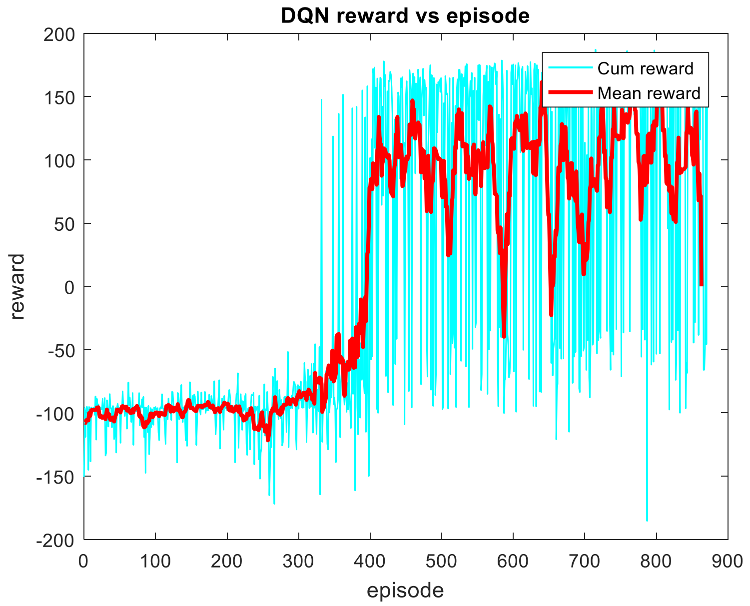
Platform			
Ubuntu16.04 + Python 3.5 + CUDA + CuDnn + GTX1080TI			
Parameter	**Value**	**Parameter**	**Value**
Max Episode	200	**Reward Discount**	0.9
Max Episode Step	200	**Memory Capacity**	3000
Learning Rate	0.001	**Batch Size**	32

Layer	Output Shape	Parameter.No
Con2d_1	(None, 32, 7, 25)	544
Con2d_2	(None, 15, 3, 64)	14,464
Con2d_3	(None, 15, 3, 64)	4,160
Flatten	(None, 2880)	0
Dense_1	(None, 512)	1,475,072
Dense_2	(None, 3)	1,539
Total		1,495,779

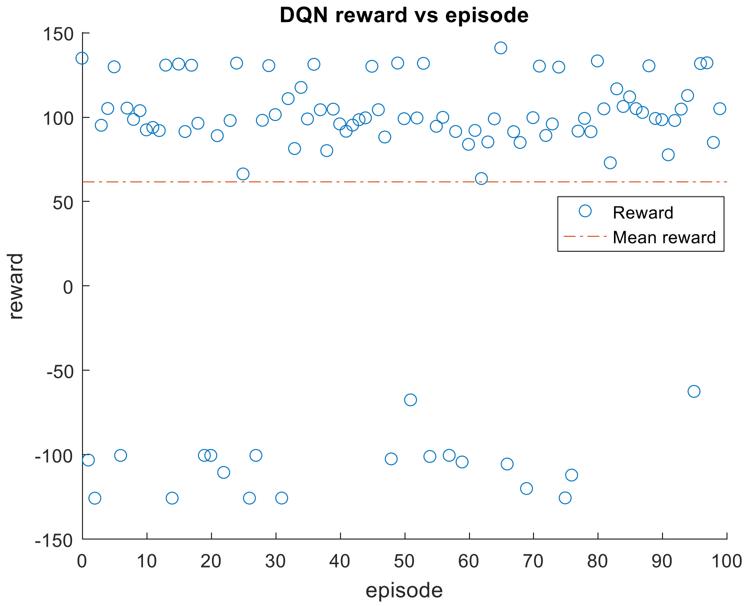
• 3.4 Vision-based Navigation Using DQN

➢ 4) DQN Result

- Training process and evaluation



Training



Evaluation

Blue line: reward at each episode
Red line: average reward every 100 episode

Success (reward>0): 80
Failure (reward<0): 10

CONTENTS

• 4.1 Policy-based Reinforcement Learning

- Unlike value-based method, policy-based RL works directly on policy.

$$\pi_\theta(a|s) = P[a|s,\theta]$$

- Define objective function as total expected future reward

$$L(\pi_\theta) = \int_S \rho^\pi(s) \int_A \pi_\theta(a|s) r(s,a) dads$$
$$= E_{s \sim \rho^\pi, a \sim \pi_\theta}\left[\left(r(s,a)\right)\right]$$

- Optimise objective end-to-end by SGD

$$\nabla L(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) dads$$
$$= E_{s \sim \rho^\pi, a \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]$$

4.2 Deep Deterministic Policy Gradient (DDPG)

- Deep: use deep neural network to represent action-value function $Q^{\mu}(s,a)$ and deterministic policy $\mu_\theta(s)$
- Deterministic: output is deterministic under the same input $a=\mu_\theta(s)$
- Objective:

$$L(\mu_\theta)=\int_s \rho^{\mu}(s)r(s,\mu_\theta(s))ds=E_{s\sim\rho^{\mu}}\left[r(s,\mu_\theta(s))\right]$$

- Deterministic policy:

$$\nabla_\theta L(\mu_\theta)=\int_s \rho^{\mu}(s)\cdot\nabla_\theta\mu_\theta(s)\cdot\nabla_a Q^{\mu}(s,a)|_{a=\mu_\theta(s)}ds$$

$$=E_{s\sim\rho^{\mu}}\left[\nabla_\theta\mu_\theta(s)\cdot\nabla_a Q^{\mu}(s,a)|_{a=\mu_\theta(s)}\right]$$

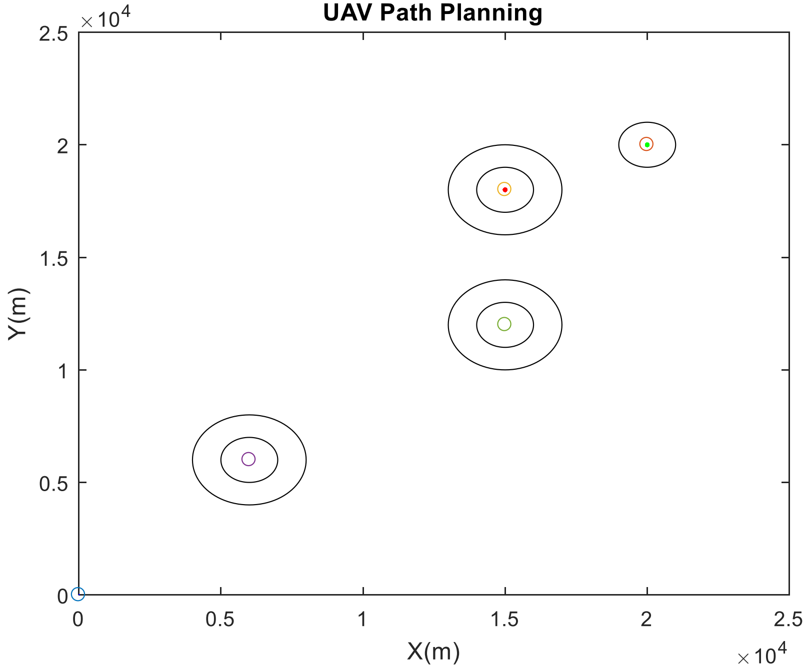
Lillicrap, et al. "Continuous Control with Deep Reinforcement Learning. " *arXiv:1509.02971*(2015).

- Off-policy: Actor is stochastic policy to ensure enough exploration

  Critic is deterministic policy to evaluate the action

• 4.3 Planar Navigation Using DDPG

➢ 1) Environment



Regarding the UAV as a mass point, employ reinforcement learning method to enable UAV to find a free-collision path toward target.

- Navigation task: （0km, 0km）>>>>>>> （20km, 20km）

  $v_{obstacle} = 200m/s$ $v_{\text{target}} = 200m/s$

  $v_{UAV} = 500m/s$ $r_{\det ection} = 1000m$

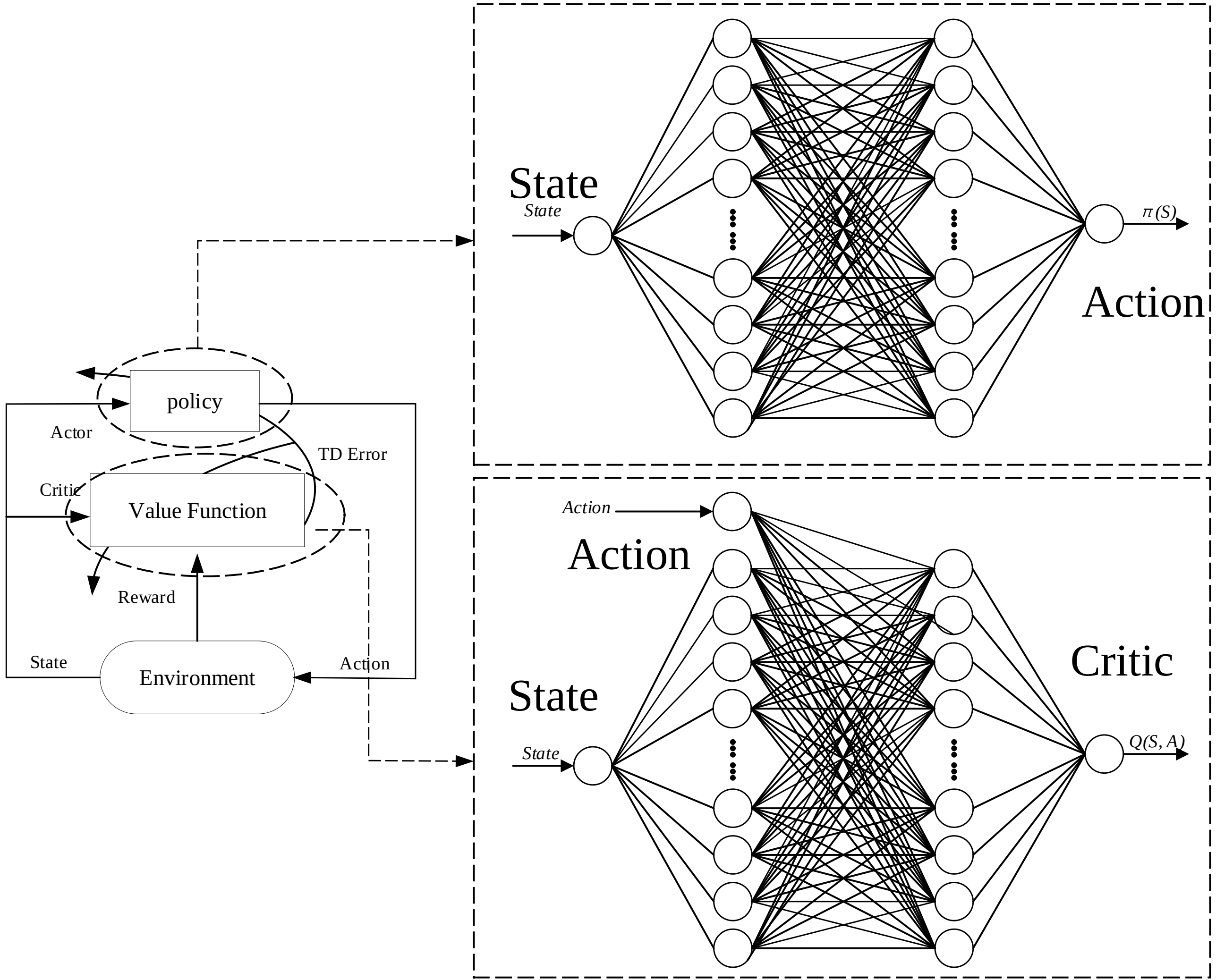
- State representation: $s_t = \left(x_m(i), y_m(i), \theta_m(i), d_{ro}(i), d_{rt}(i)\right)$

- Action space: $a_t \in \left[-45^\circ, +45^\circ\right]$

- Reward function: $r = \alpha\left(d_{rt} - d_{rt}'\right) - \beta / d_{ro} - \varphi \cdot \theta$

• 4.3 Planar Navigation Using DDPG

➢ 2) DDPG Network Architecture



Platform			
Ubuntu14.05 + Python3.5			
Parameter	**Parameter**	**Value**	**Value**
Max Episode	200	**Memory Capacity**	3000
Max Episode Step	200	**Batch Size**	32
Learning Rate Actor	0.001	**Replace Iter Actor**	800
Learning Rate Critic	0.0001	**Replace Iter Critic**	700
Reward Discount	0.9		

Actor

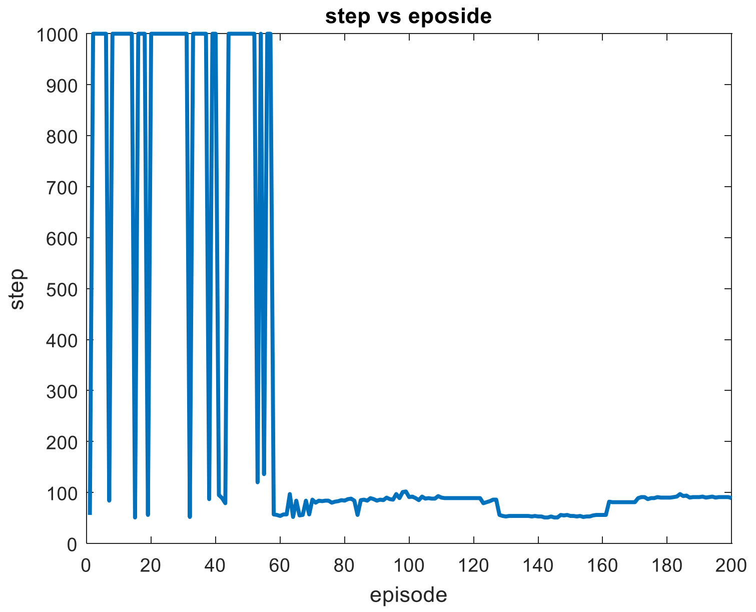
Layer	Output Shape	Paramet er.No
Input	(None, 7)	
Dense_1	(None, 100)	800
Dense_2	(None, 20)	2020
Dense_3	(None, 1)	21
Total		2841

Critic

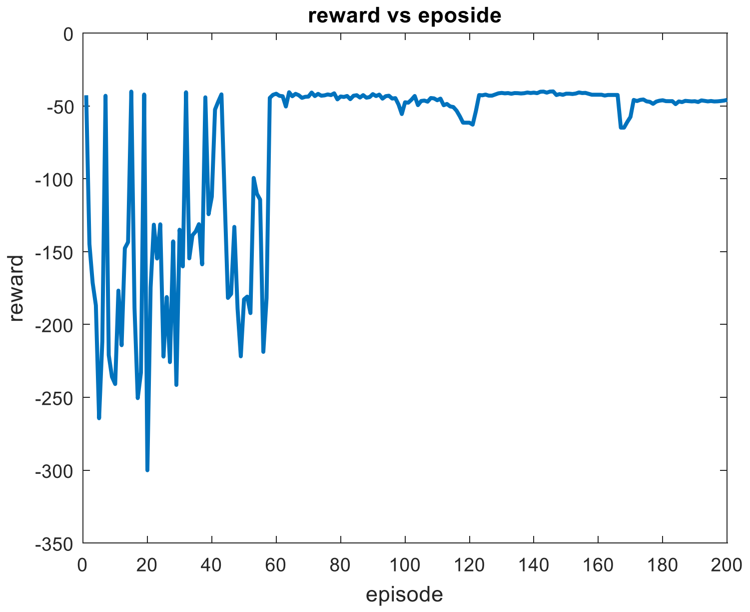
Layer	Output Shape	Parame ter.No
Input_state	(None, 7)	
Input_action	(None, 1)	
Concatenate	(None, 8)	
Dense_1	(None, 100)	900
Dense_2	(None, 20)	2020
Dense_3	(None, 1)	21
Total		2940

• 4.3 Planar Navigation Using DDPG

➢ 3) Result



Steps in training



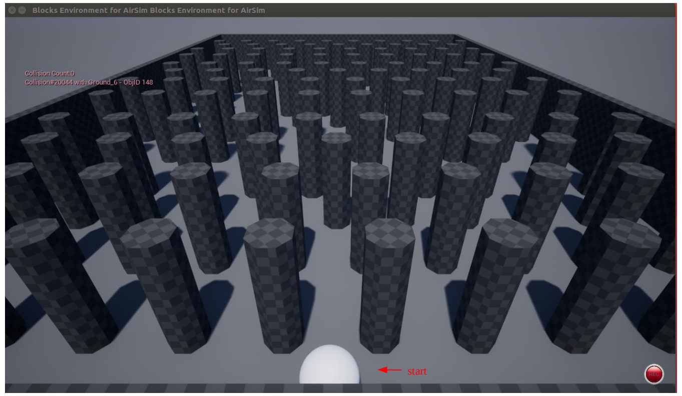
Cumulative reward in training

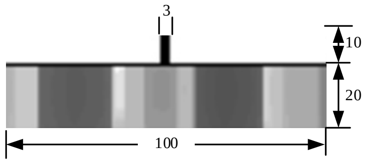
• 4.4 Vision-based Navigation Using DDPG

➢ 1) Environment

- Navigation Task: UAV has to reach the goal in minimum amount of time without colliding with any obstacle.

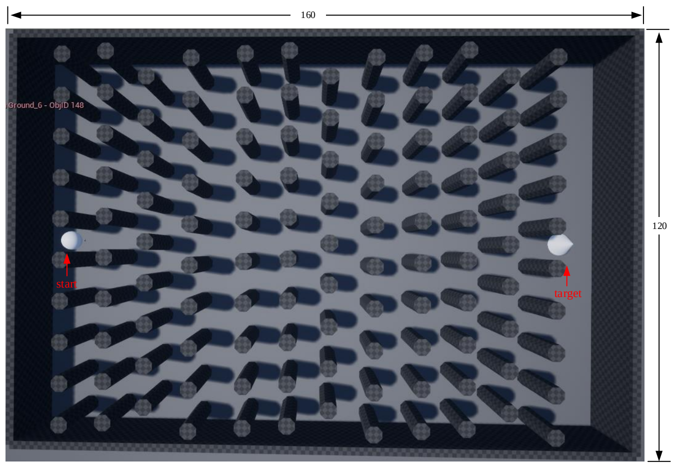
(10m, 0m) >>>>>> (150m, 0m)





- State representation

- Action space $a_t \in [-1, +1] \longrightarrow$ Yaw angle $\theta_t = a_t \times 30^\circ / s$



- Reward function

$$r = \begin{cases} 100 & if \quad success == TRUE \\ -100 & if \quad collided == TRUE \\ -1 + \Delta d_{t-1} - \Delta d_t - |\phi| & otherwise \end{cases}$$

• 4.4 Vision-based Navigation Using DDPG

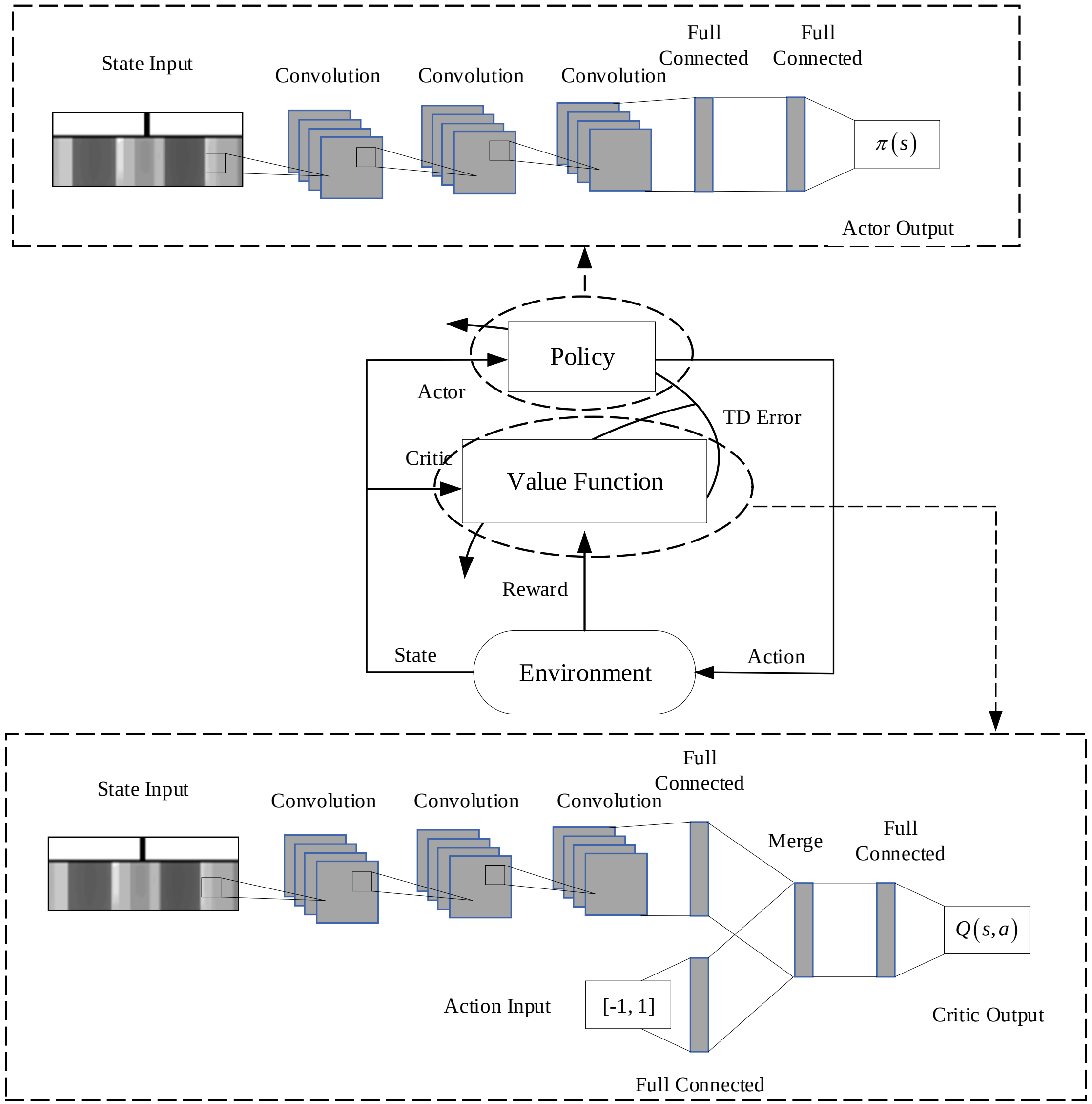
➢ 2) DDPG Network Architecture

Actor network

Input state, connect to 3 CNN and 2 FC
Output action

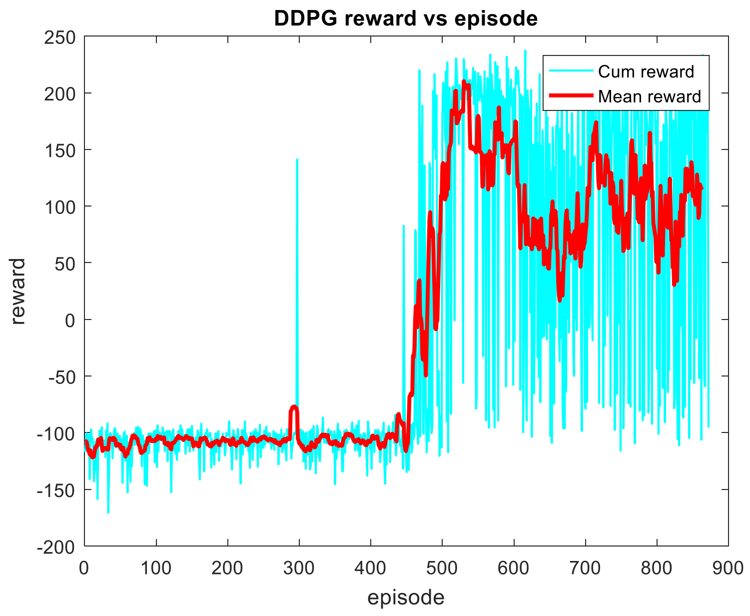
Critic network

Input state, connect to 3 CNN
Input action, connect to FC and merge with state information
Output action-value function to critic the action

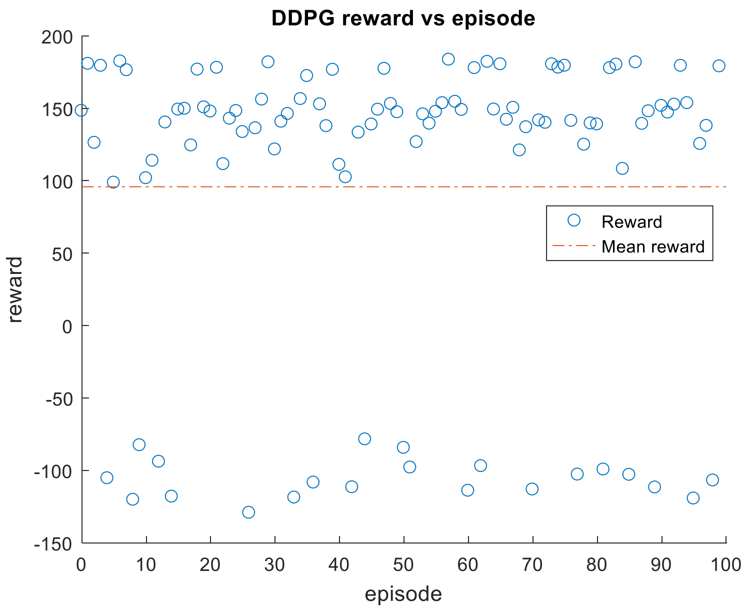


• 4.4 Vision-based Navigation Using DDPG

➢ 3) DDPG Result

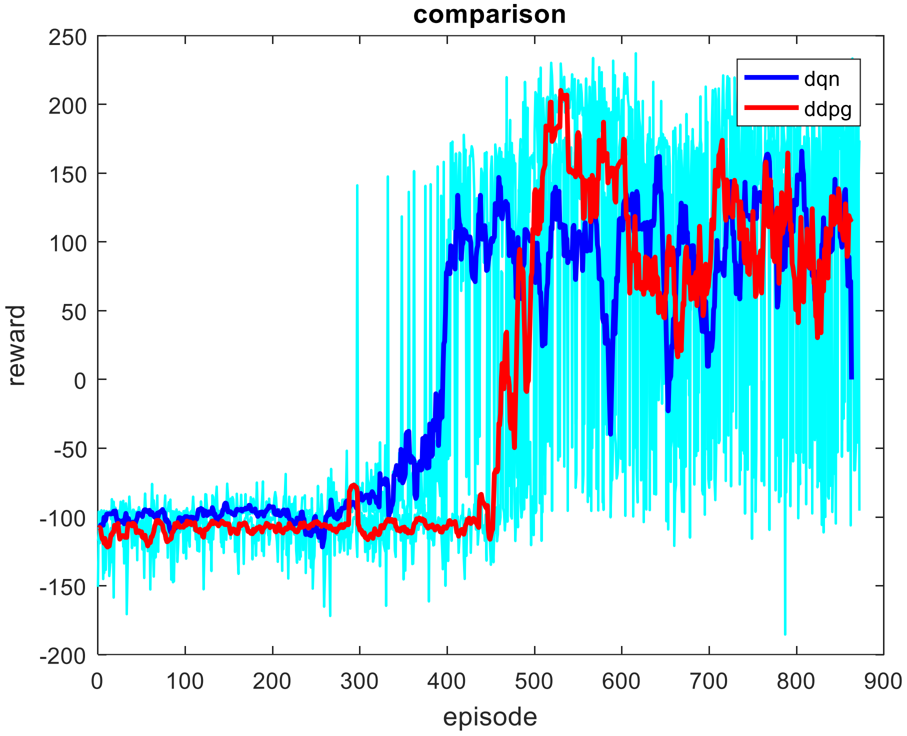


Training Process



Evaluation

- **4.5 Comparison**
  - Training process and evaluation



Training Process

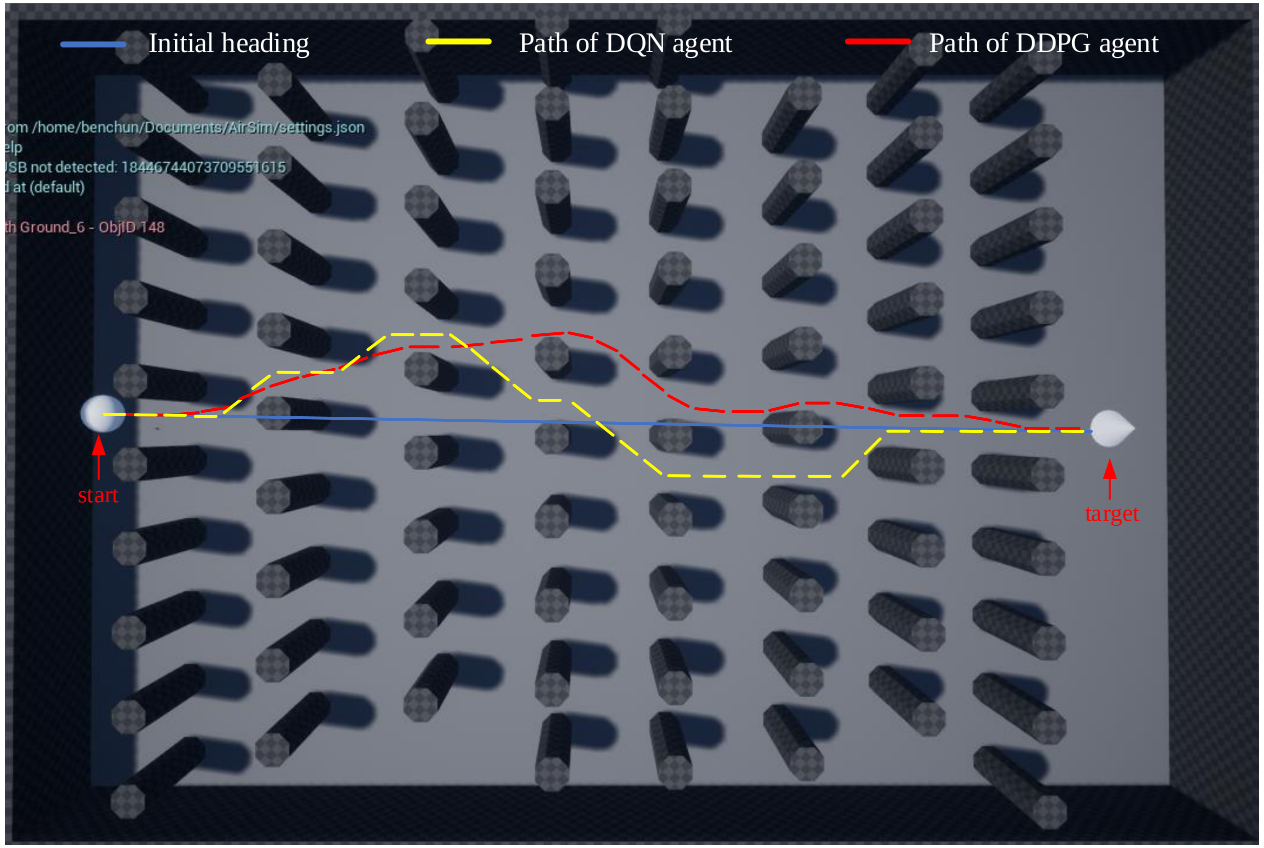
	DDPG Agent	DQN Agent
Success rate	79%	80%
Mean reward	98.6	61.6

Evaluation

- **4.5 Comparison**

  - Path and action

    - DQN Agent: able to reach the goal
    - DDPG Agent: path is much smoother



CONTENTS

• 5.1 Conclusion

We applied deep reinforcement learning on vision-based autonomous navigation within a 3D simulated environment.

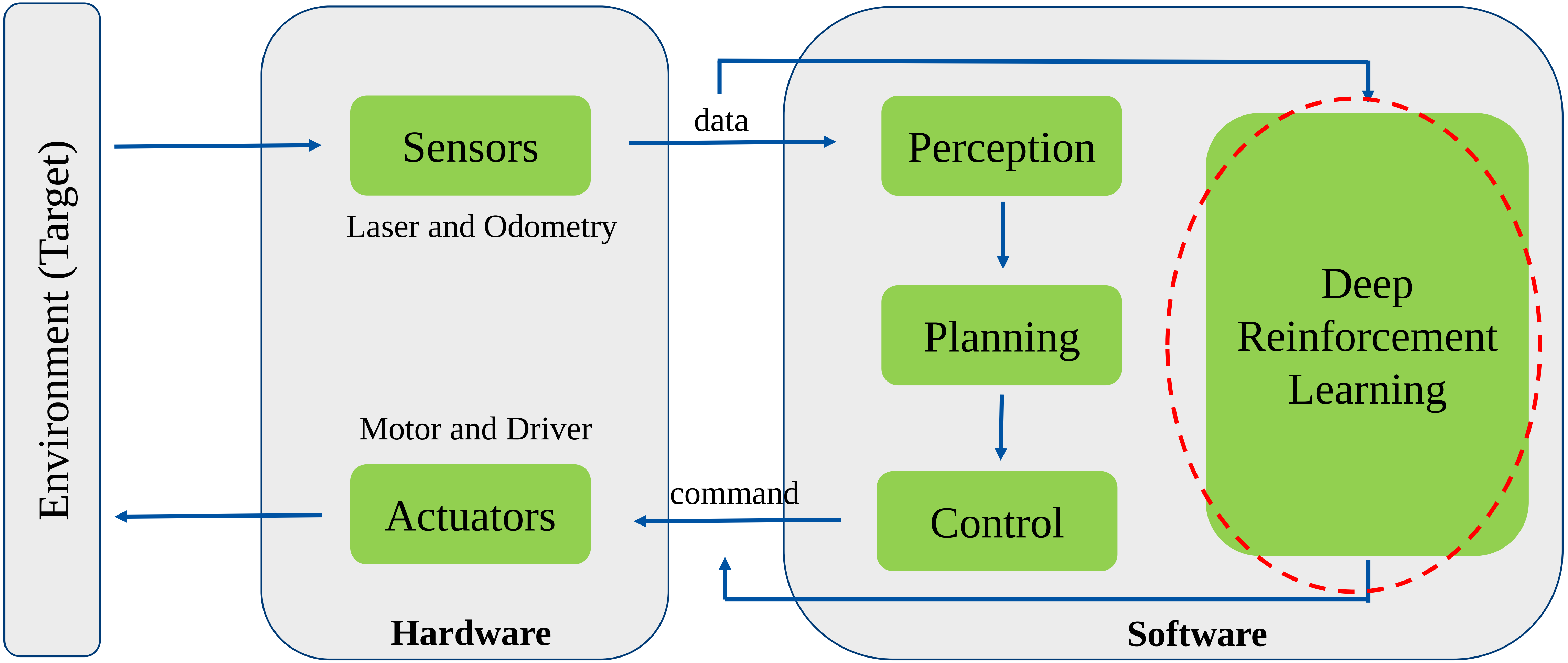
Formulated the navigation task as a Partially Observable Markov Decision Process (POMDP)

Employed deep Q-network (DQN) algorithm to calculate the estimated values of three discrete actions and then select actions to maximise cumulative reward.

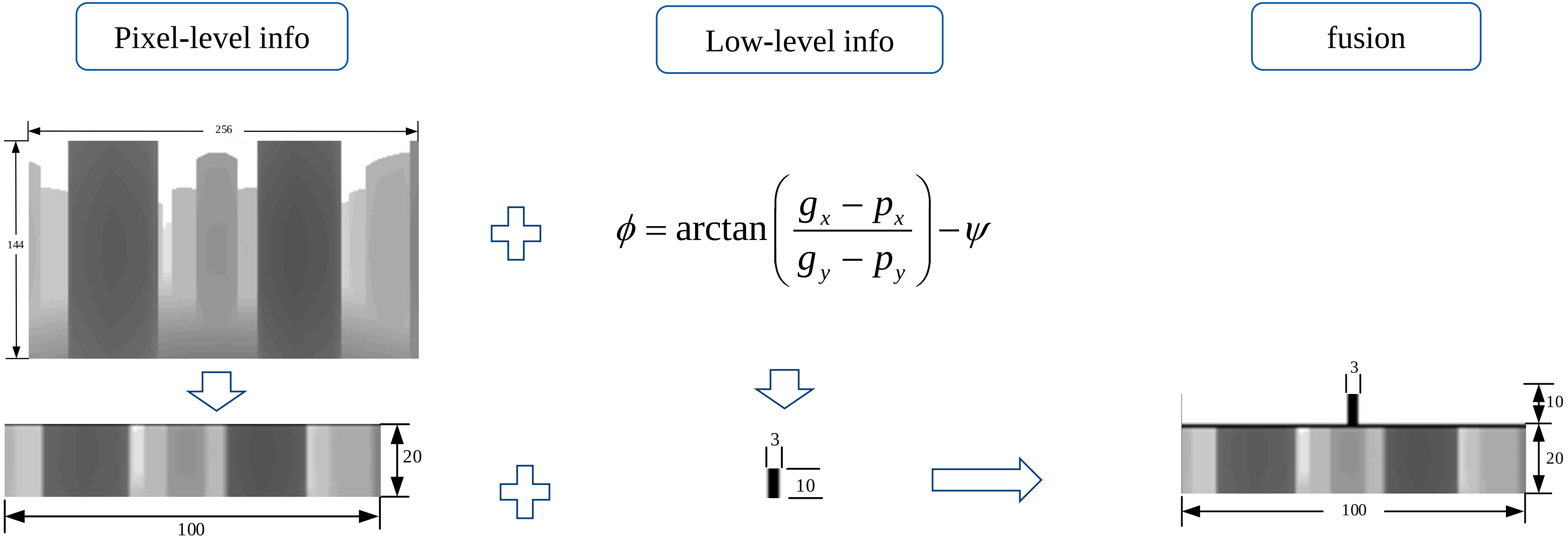
Extended deep deterministic policy gradient (DDPG) algorithm with convolutional neural network to deal with depth image and enable UAV to act in continuous action space.

Demonstrated the validation of this approaches AirSim simulator

- **5.2 Advantages**
  - ➢ 1) Apply DRL in autonomous navigation

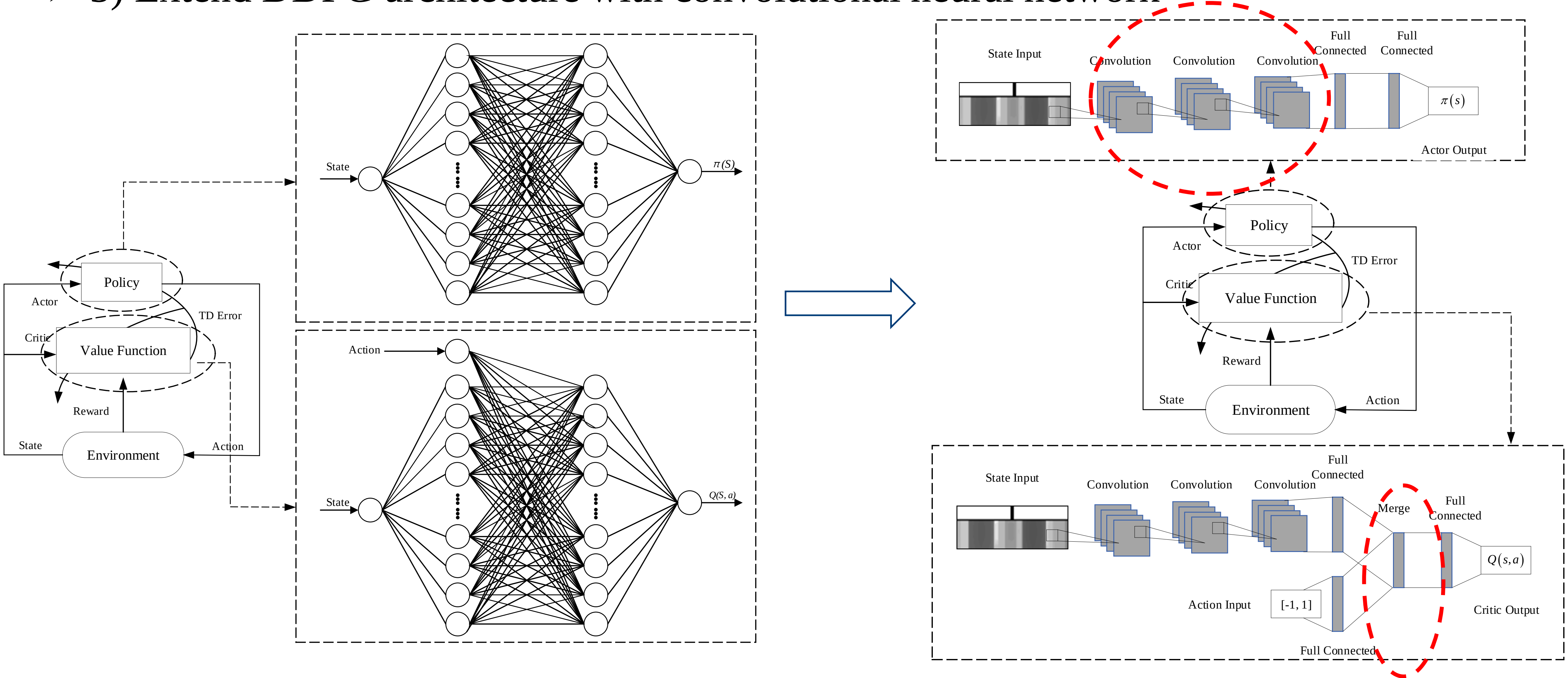


- **5.2 Advantages**
  - ➢ 2) Fuse image information with target information



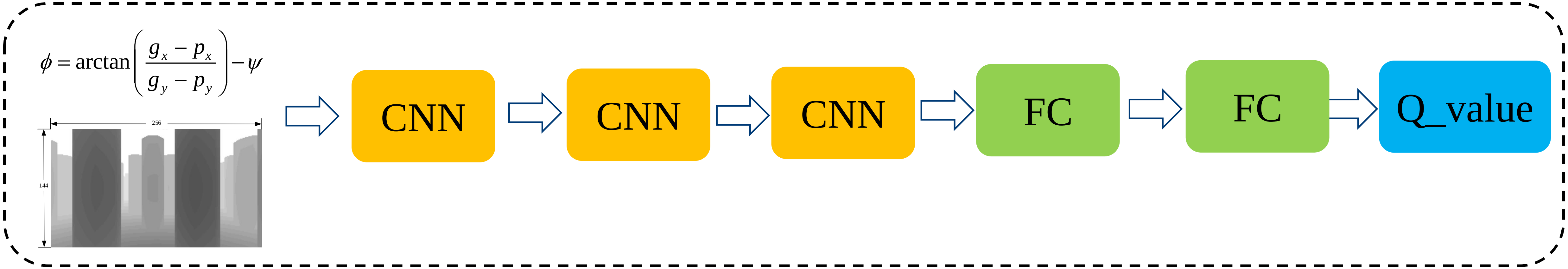
- **5.2 Advantages**

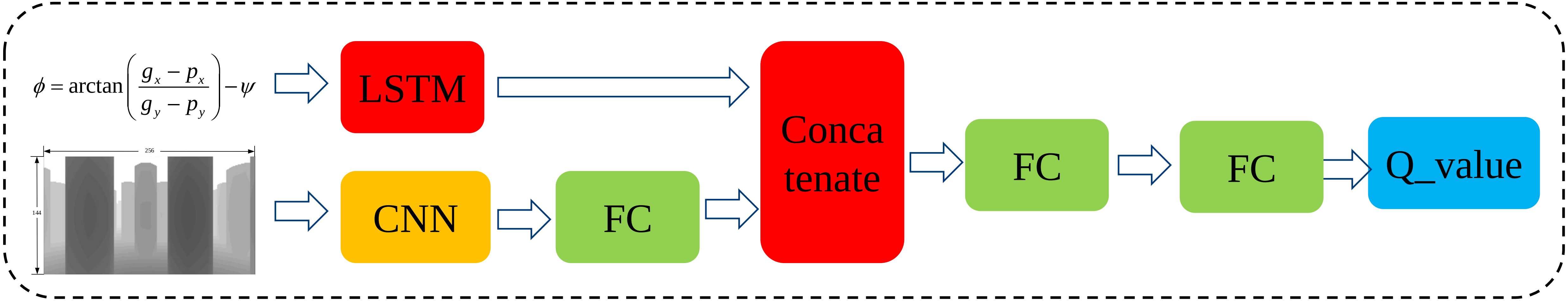
  ➢ 3) Extend DDPG architecture with convolutional neural network



- **5.3 Future Work**
  - ➢ 1) Data fusion in DQN

Introduce Long Short Term Memory (LSTM) to deal with low-level info

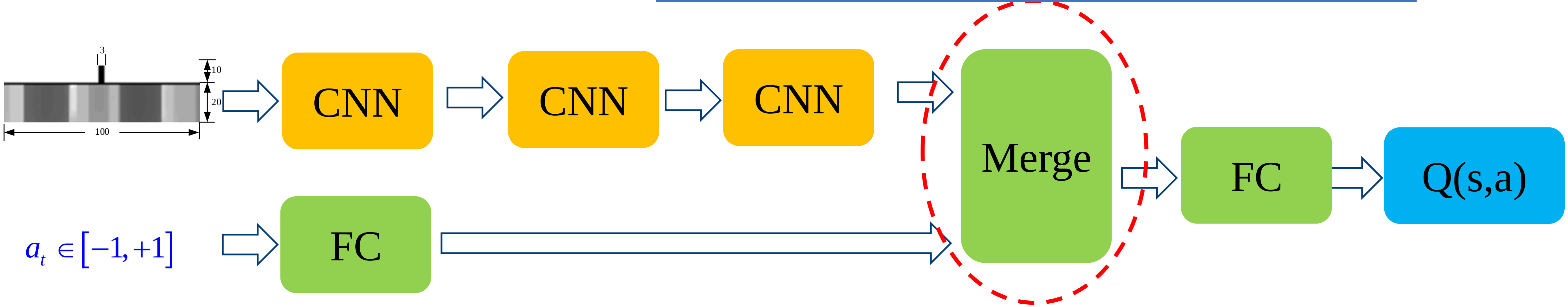


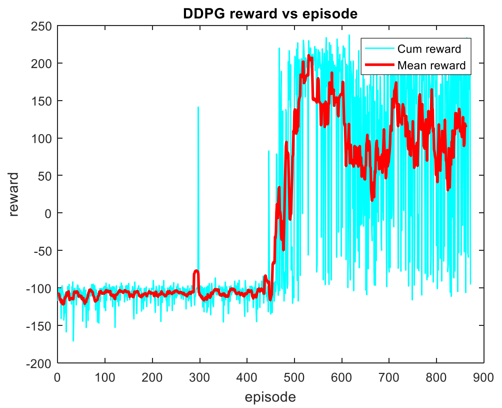


- **5.3 Future Work**

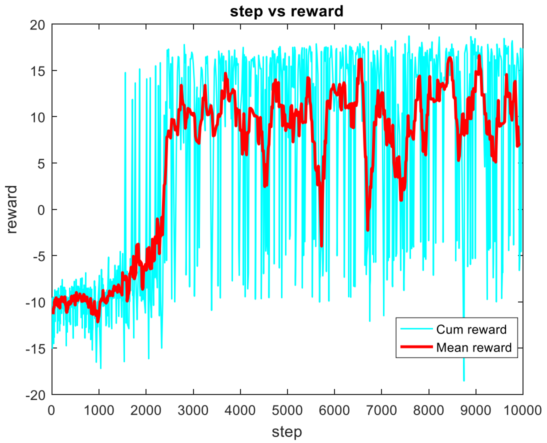
  - ➢ 2) Improve robust of DDPG

Find a better solution to merge state information and action information

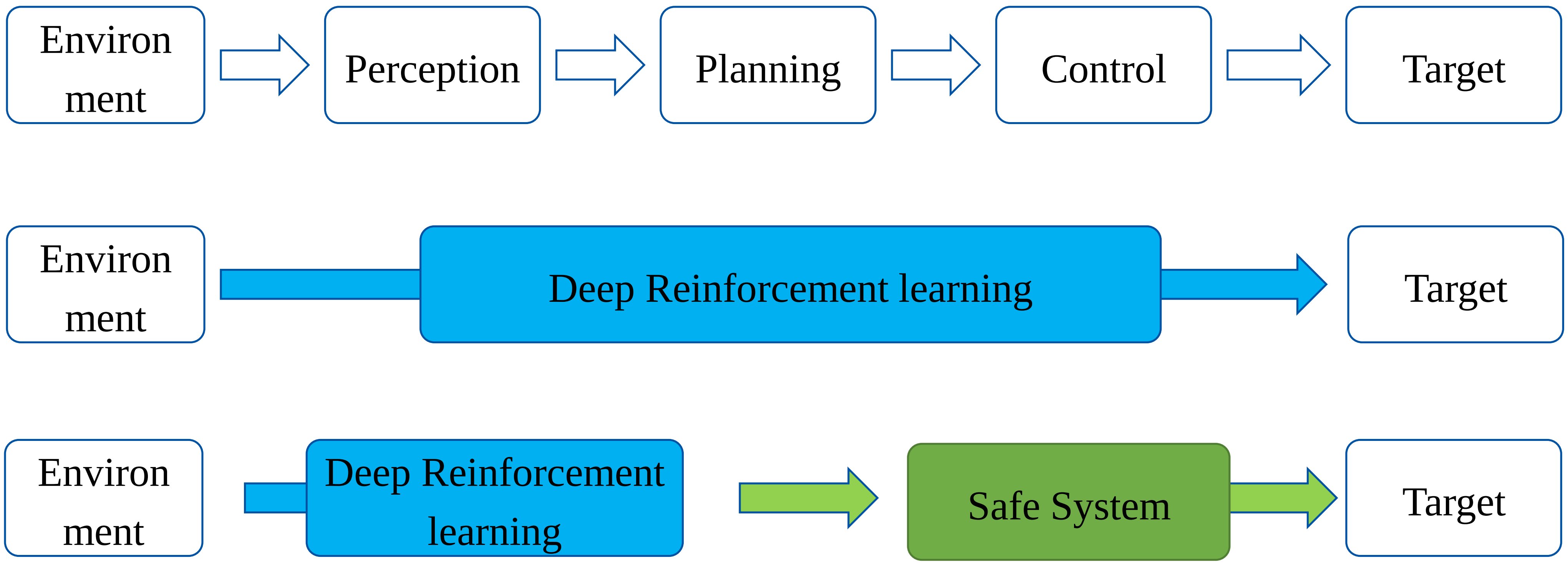




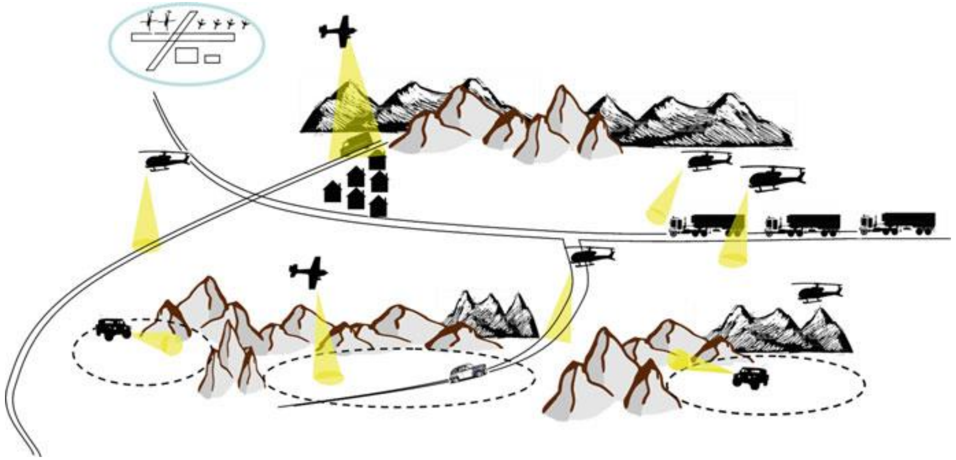
VS



- **5.3 Future Work**
  - 3) Safe reinforcement learning ⇨ Add safe system to avoid failure



- **5.3 Future Work**
  - 4) Generalization capability and practicality ⇨ Transfer to different environment and real world



CONTENTS

6. Future

- **Research Interests**

- Autonomous System
- Robotics
- UAV Navigation

6. Future

- **Project experience**

* Master Thesis	[Autonomous Navigation of UAV Using Deep Reinforcement Learning]
* Project	[End-to-end Learning in Motion Planning for Robots]
* Survey	[Application of End-to-end Learning Method]
* Project	[Multi UAV Navigation]
* Internship	[Autonomous Robotics Motion Planning]
* Survey	[Autonomous Driving Framework Using Traditional Method]
* Comparison	[Difference about Robotics UAV and Vehicles]

THANKS YOU!