Autonomous Navigation of UAV Using Deep Reinforcement Learning

School of Automation Science and Electronic Engineering
Beihang University

- **Abstract**

Autonomous navigation of unmanned aerial vehicles (UAV) can be seen as a process that robots make a plan how to safely and quickly reach the target location. Traditional navigation methods rely heavily on environment information and manual design, making them difficult to adapt new environment. In the paper, we aim to apply deep reinforcement learning (DRL) on autonomous navigation, learning optimal control policy through interaction with environment.

The main work of the paper was shown as follows: 1) Theoretical basis of deep reinforcement learning was given. 2) Vision-based autonomous navigation task using Deep Q-Network method was discussed. 3) Deep Deterministic Policy Gradient was improved to solve the same problem, enabling the UAV to take depth image as input and act in continuous space. 4) Simulations in AirSim support the idea of autonomous navigation through DRL methods.

Keywords: *deep reinforcement learning*, *autonomous navigation*, *depth image*, *continuous action*, *simulation*

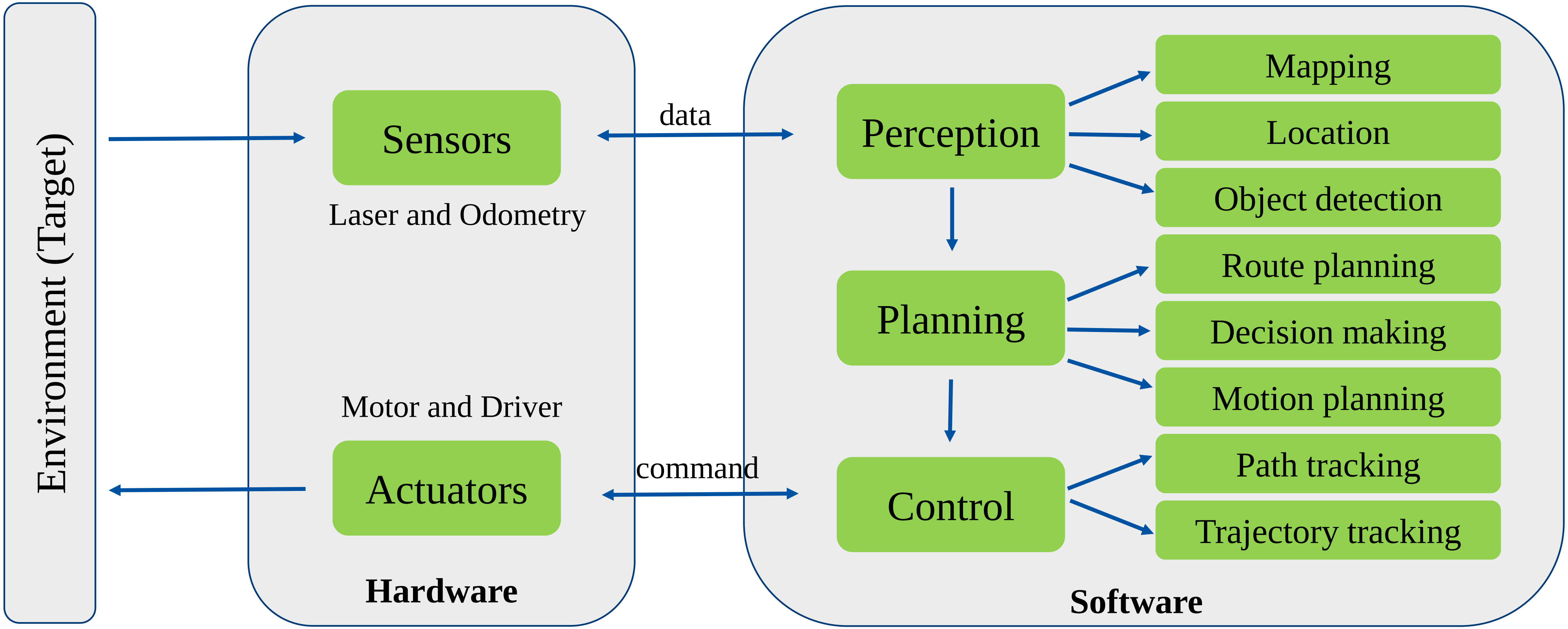
CONTENTS

CONTENTS

1. Introduction

- **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

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



1. Introduction

- **Autonomous UAV Navigation**
- Challenges

Perception

Manual designed features are inaccurate

Path planning

Hard-coded path planning is inefficient

Control

Demands for sophistic behavior increase

Traditional navigation method rely heavily on human developer to decomposition feature manully and design navigation system

More intelligent methods need to be developed to core with more complex environment

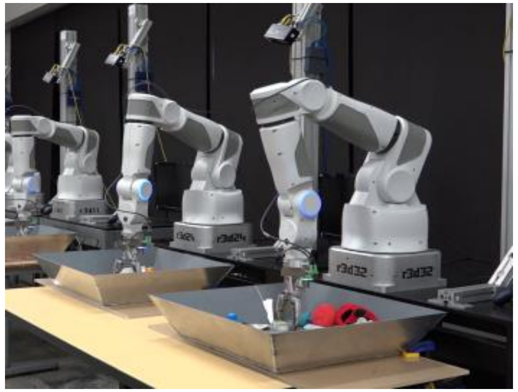
1. Introduction

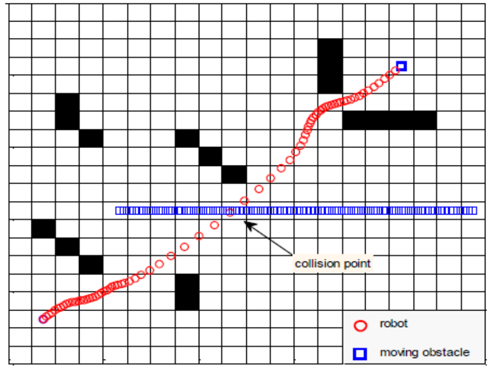
- **Deep Reinforcement Learning**

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

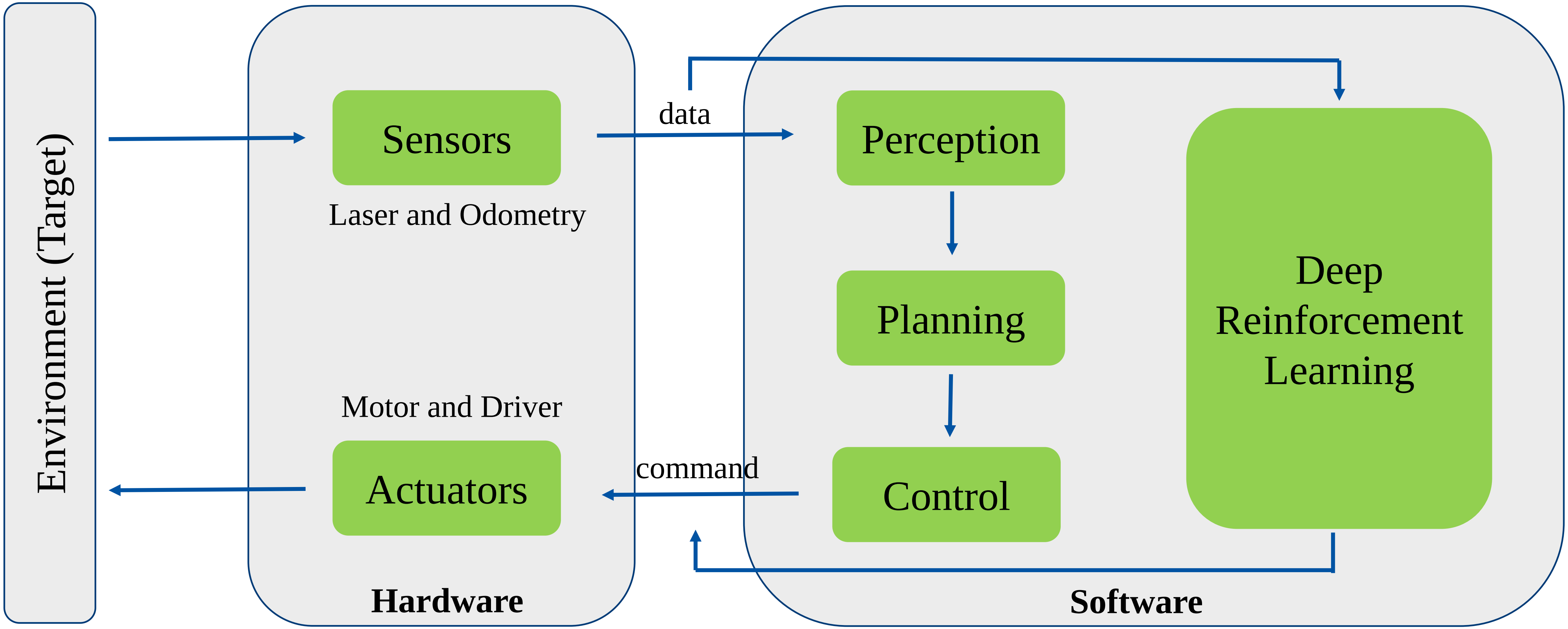
Applications

- Play Games: Atari, Go, …
- Explore worlds: 3D worlds, Labyrinth, …
- Control physical system: manipulate, walk, …
- Interact with users: recommend, optimise, …





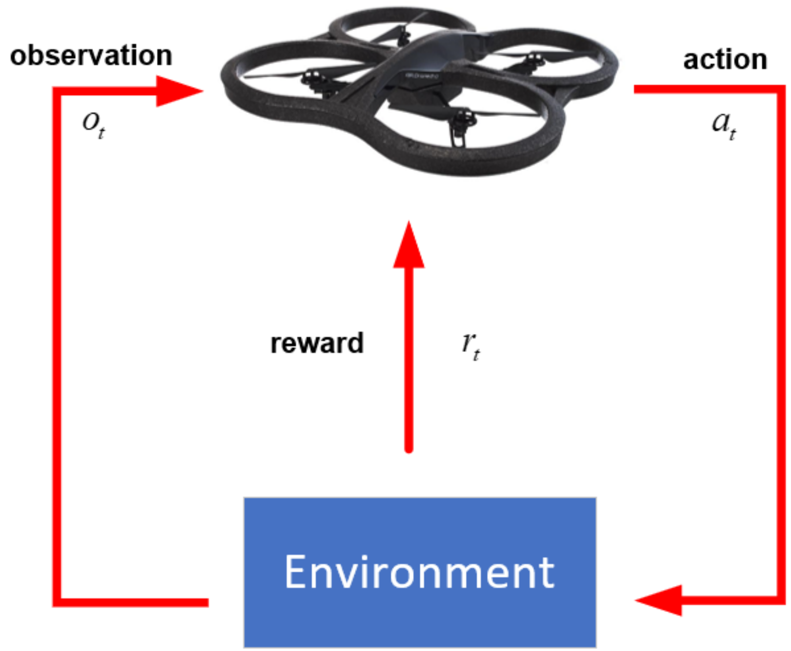
- **Deep Reinforcement Learning**



CONTENTS

2. Background

- **Agent and Environment**



- 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}

- Policy π is a behaviour function selecting actions given states

$$a = \pi(s)$$

- Value function $Q^{\pi}(s,a)$ is expected future reward

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

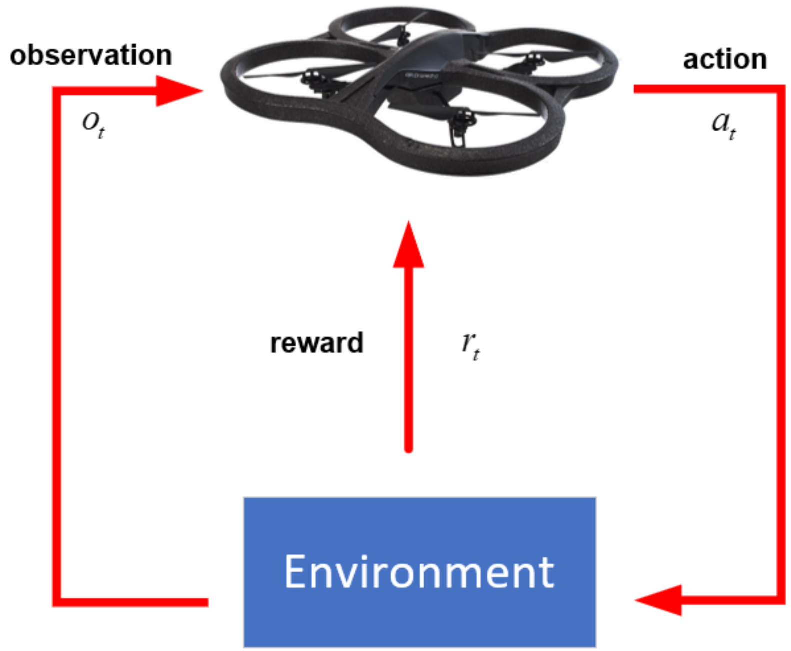
- Value function decomposes into a Bellman equation

$$Q^{\pi}(s,a) = \mathrm{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a') \mid s,a\right] \longrightarrow \text{Policy-based RL}$$

- Optimal value function selects maximum value over all decisions

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

- **Agent and Environment**



- Value-based RL
  - ➢ Estimate the optimal value function $Q^*(s,a)$
  - ➢ This is the maximum value achievable under any policy
- Policy-based RL
  - ➢ Search directly for the optimal policy π^*
  - ➢ This is the policy achieving maximum future reward

• Value-based Reinforcement Learning

- 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)

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

2. Background

- **Value-based Reinforcement Learning**

- Deep Q-Network

DQN provides a stable solution to deep value-based RL

1.Use experience replay

- Break correlations in data
- Learn from all past policies
- Using off-policy Q-learning

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

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

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

2. Freeze target Q-network

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

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

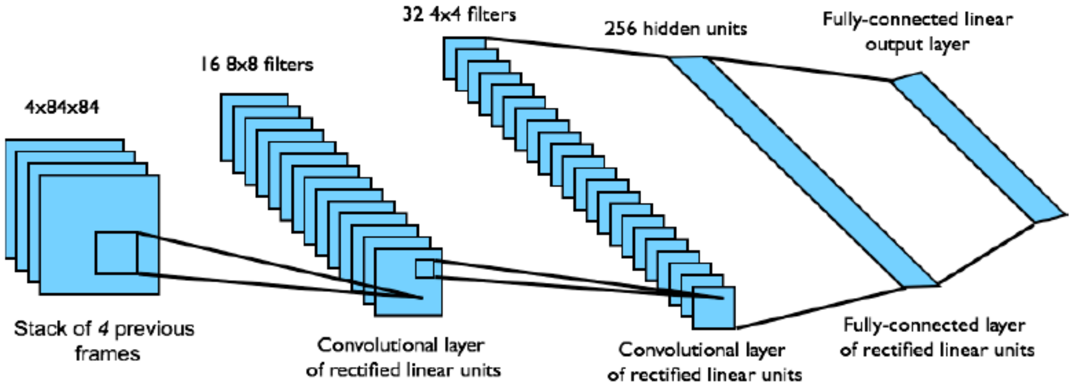
3. Clip rewards or normalize network adaptively to sensible range

- Robust gradients

- **Value-based Reinforcement Learning**

- Deep Q-Network in Atari

Mnih, V, et al. "Playing Atari with Deep Reinforcement Learning." *Computer Science* (2013).
Mnih, V, et al. "Human-level Control Through Deep Reinforcement Learning. " *Nature* 518.7540(2015):529.



• Policy-based Reinforcement Learning

- Represent policy by deep network $a = \pi\left(s, u\right)$ with weights u
- Define objective function as total expected future reward

$$J\left(u\right) = \mathrm{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots\right]$$

- Optimise objective end-to-end by SGD
  - i.e. Adjust policy parameters u to achieve more reward
- The gradient of the policy is given by

$$\frac{\partial J\left(u\right)}{\partial u} = \mathrm{E}_s\left[\frac{\partial Q^{\pi}\left(s,a\right)}{\partial u}\right] = \mathrm{E}_s\left[\frac{\partial Q^{\pi}\left(s,a\right)}{\partial a}\frac{\partial \pi\left(s,u\right)}{\partial u}\right]$$

- Policy gradient is the direction that most improves Q

- **Policy-based Reinforcement Learning**
  - Deep Deterministic Policy Gradient

DDPG = Actor-Critic + DQN

- Actor is a policy $\pi(s,u)$ with parameters u: $s \underset{u_1}{\rightarrow} \dots \underset{u_n}{\rightarrow} a$
- Critic is value function $Q(s,a,w)$ with parameters w

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

- Critic estimates value of current policy by Q-learning

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

- Actor updates policy in direction that improves Q

$$\frac{\partial J(u)}{\partial u} == \mathrm{E}_s\left[\frac{\partial Q^{\pi}(s,a,w)}{\partial a}\frac{\partial \pi(s,u)}{\partial u}\right]$$

1.Use experience replay for both actor and critic

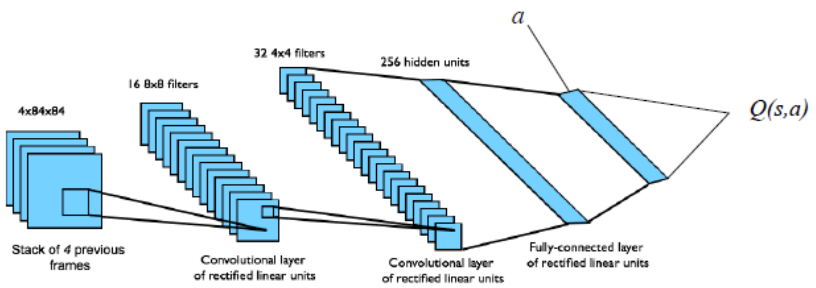
2. Freeze target Q-network to avoid oscillations

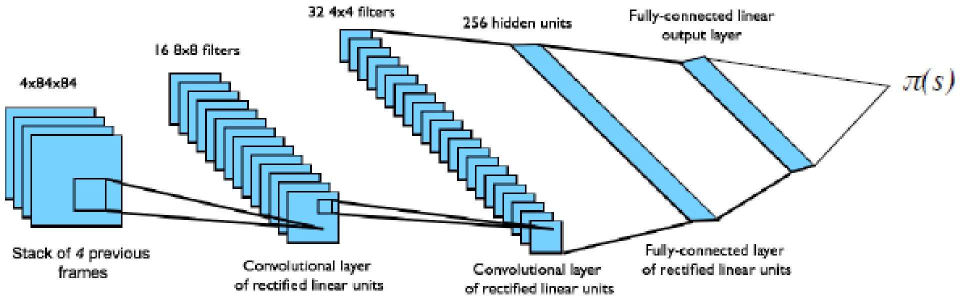
3. Clip rewards or normalize network adaptively to sensible range

- **Policy-based Reinforcement Learning**
  - DDPG for Continuous Control

More details in course: CS 294

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



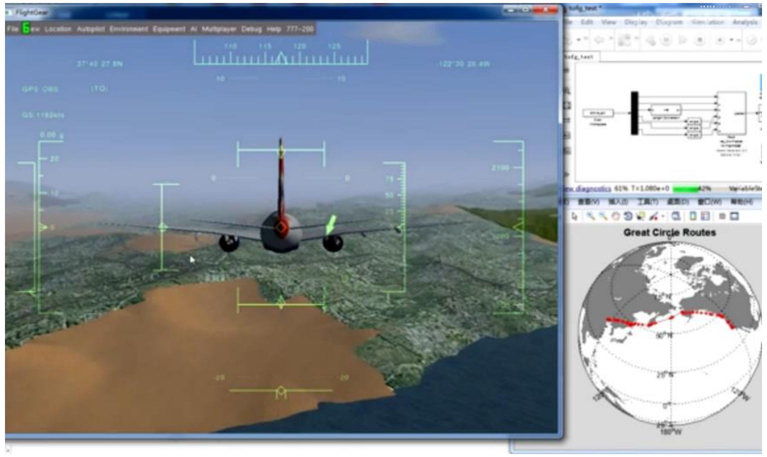


CONTENTS

3. Simulation

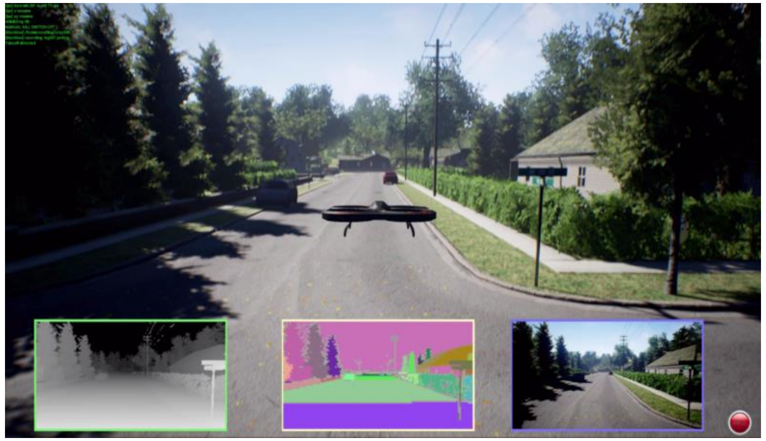
- **Tools**

- 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. Simulation

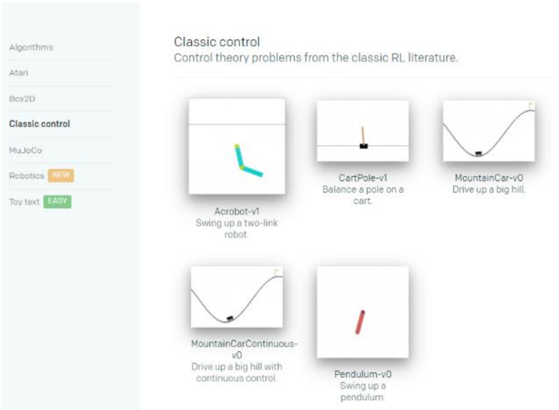
- **Tools**

- AirGym

AirSim

OpenAI Gym

Air Gym



Render the environment
Provide API to send data and receive commands

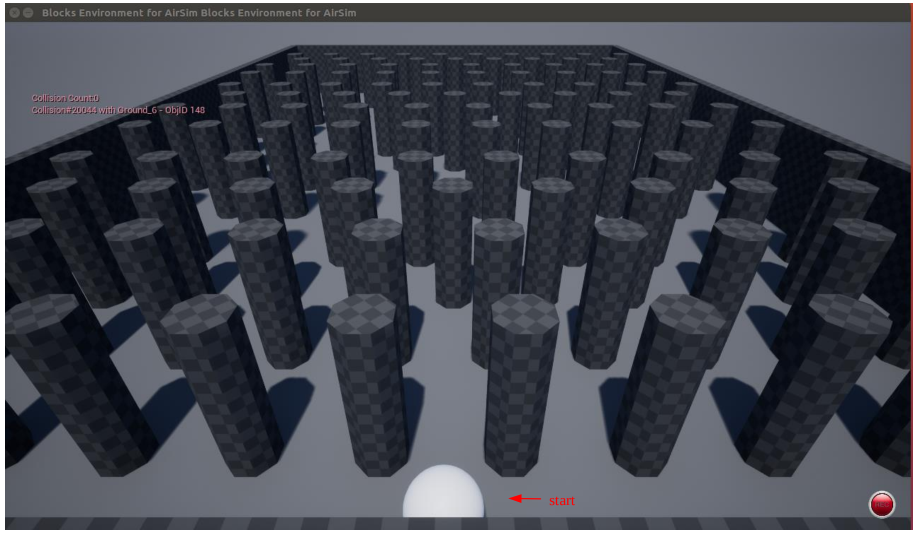
Model the navigation task
Train the agent using reinforcement learning algorithm

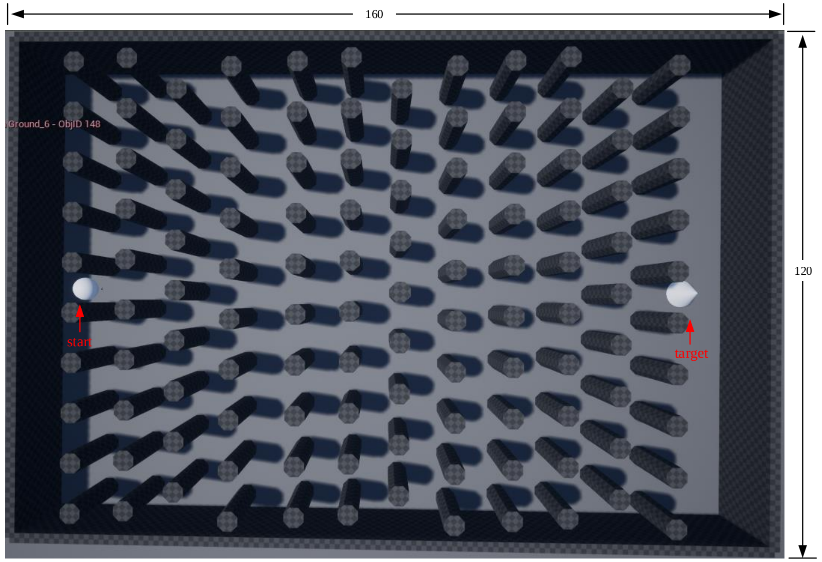
3. Simulation

- **Problem Formulation**

- AirGym-v4
  - 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

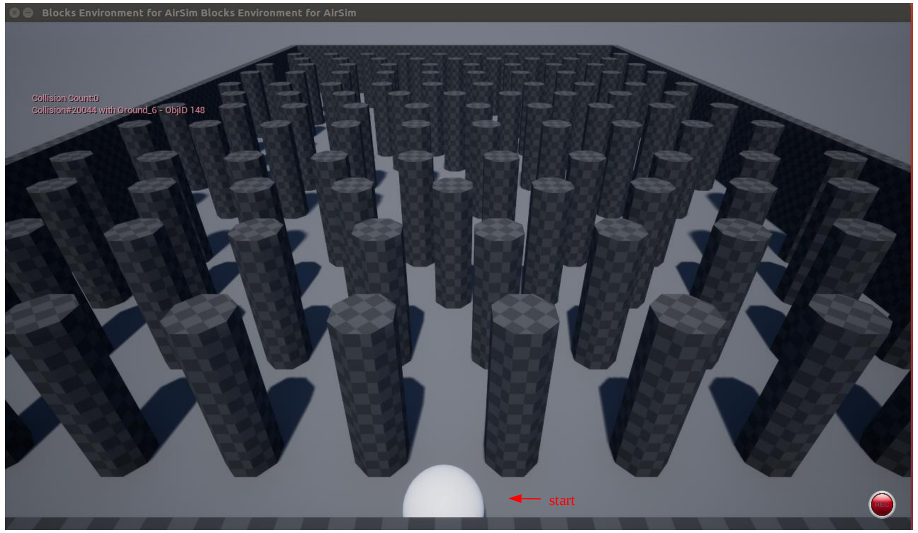


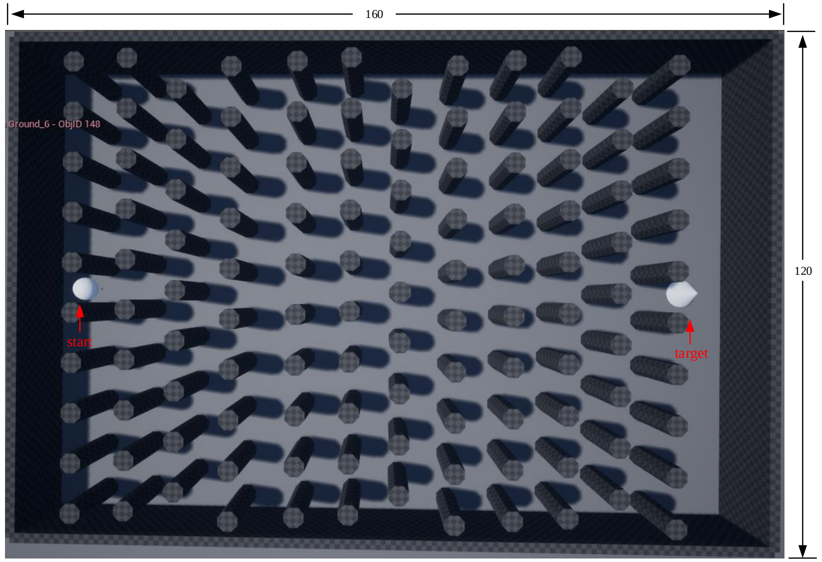


160m*120m*15m

- **Problem Formulation**

- Navigation task
  - (0m, 0m, 0m) >>>>>> (150m, 0m, 0m)
  - the UAV has to reach the goal in minimum amount of time without colliding with any obstacle.



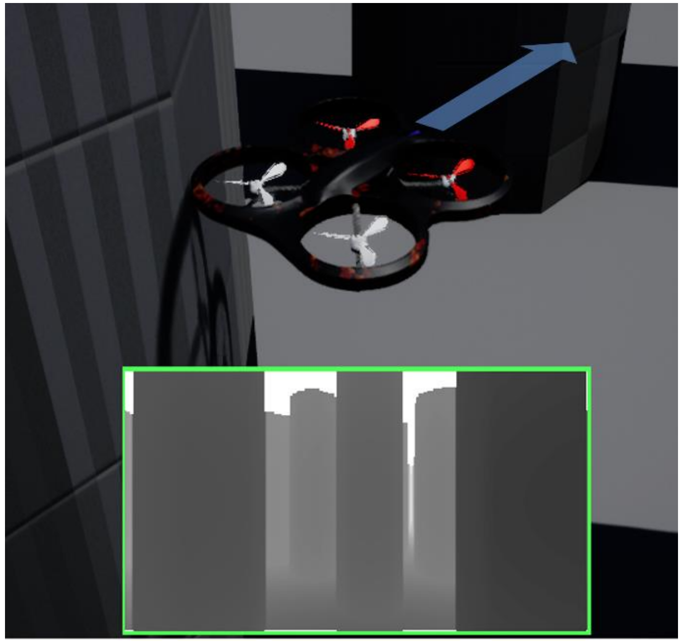


160m*120m*15m

- **Problem Formulation**

- Action Space
  - UAV flies at fixed level (6m) and at constant speed (4m/s)

- Discrete actions in DQN: $a_t \in \{-1, 0, +1\}$
  - 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°



- Continuous actions in DDPG: $a_t \in [-1, +1]$
  - Angle to rotate: $\theta_t = a_t \times 30^\circ / s$
  - E.g.: '-1': Rotate left with 30° /s for 1 s~=30°
  - '+1': Rotate right with 30° /s for 1 s~=30°

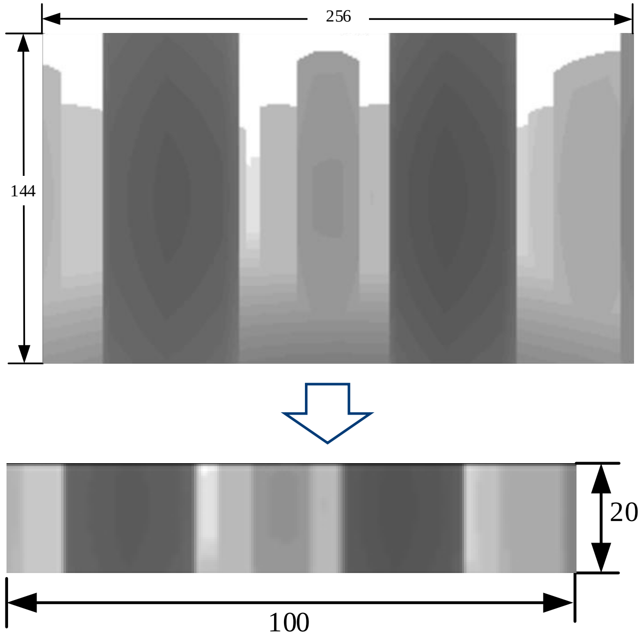
- **Problem Formulation**

- State Representation

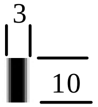
Depth image processed

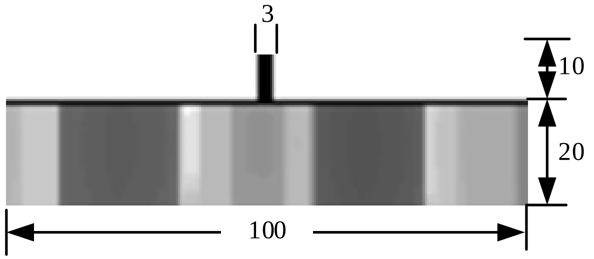
Goal information encoded

State representation



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





- **Problem Formulation**

- Reward Function

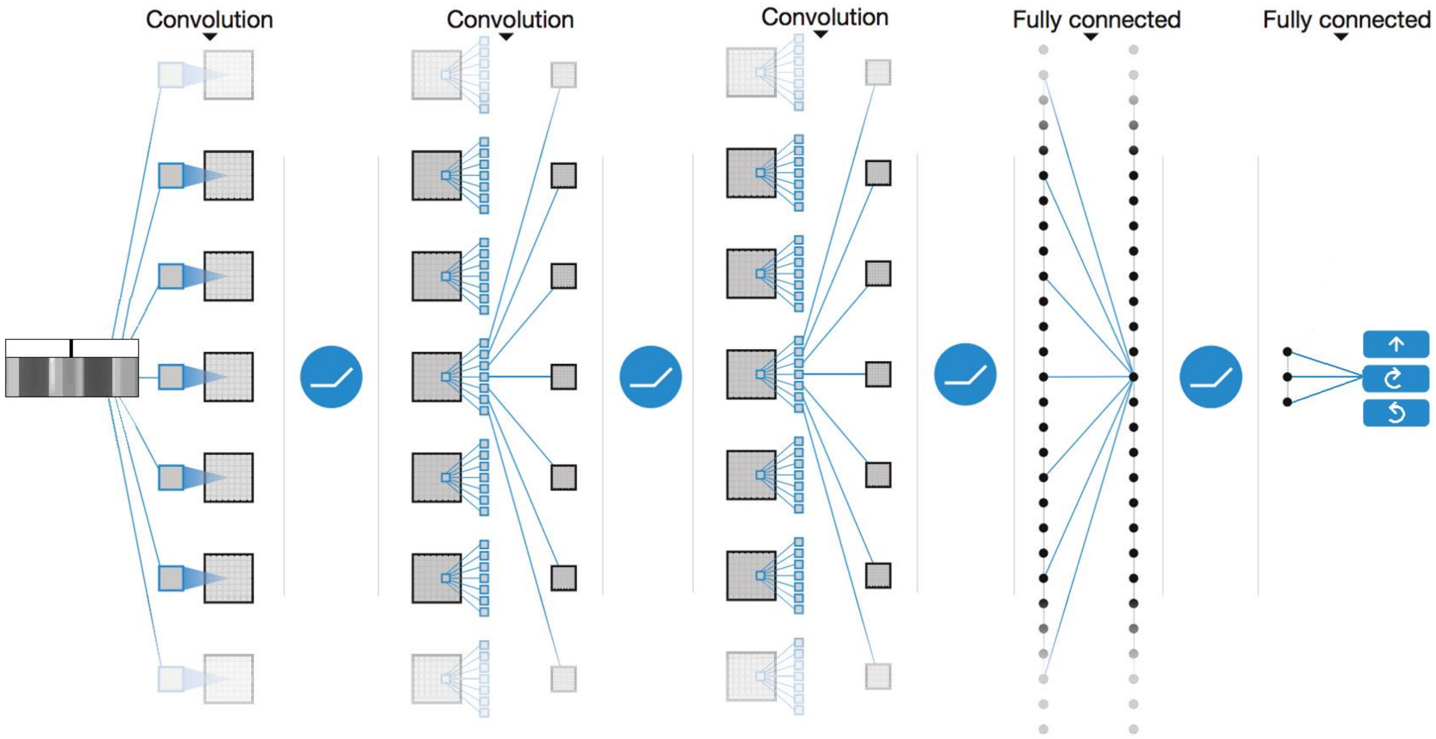
$$r=\begin{cases}10 & \text{if} \quad success == TRUE \\ -10 & \text{if} \quad collided == TRUE \\ -0.1+\Delta d_{t-1}-\Delta d_t-|\phi| & otherwise\end{cases}$$

Terminal reward	Positive when UAV reached the target Negative when UAV touched the obstacles
Time reward	Constant negative, forcing the UAV to solve task quickly
Approach reward	positive when UAV flied closer to target Negative when UAV flied further to target
Track angle reward	Negative, force the UAV directly head toward the target

CONTENTS

- **DQN Agent**

  - 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



4. Results

- **DQN Agent**

- Hyperparameters and Platform

Platform	
Ubuntu16.04 + Python 3.5 + CUDA + CuDnn + GTX1080TI	
Parameter	**Value**
Training Steps	10,000
Memory Capacity	2,000
Batch Size	32
Discount Factor	0.9
Learning Rate	0.0025
Exploitation probability	1 to 0.1

Layer	Output Shape	Parameter.No
State Input	(None, 1, 30, 100)	
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

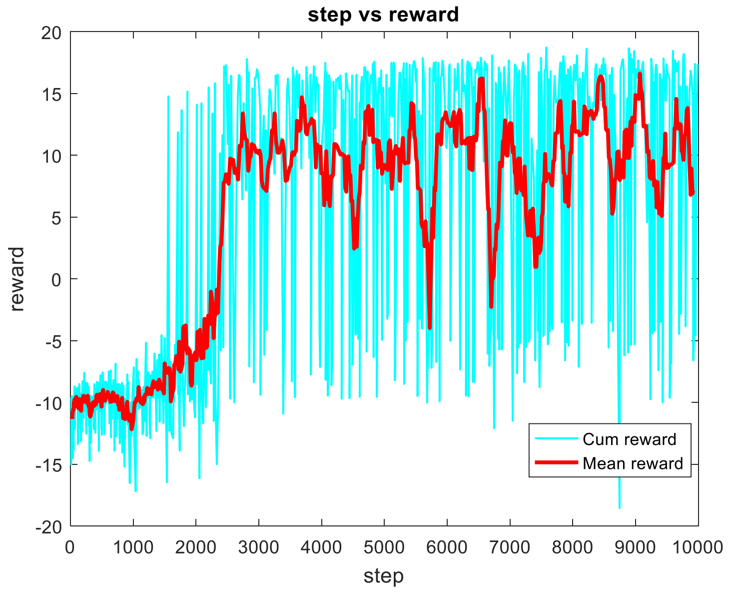
4. Results

- **DQN Agent**

- Video

- **DQN Agent**

- Results
  - The cumulative reward starts to increase after 2000 steps when the memory is full
  - There still exists failures because of the bad experience
  - Evaluation: success : 90% , average reward: 15



Training process

Evaluation

4. Results

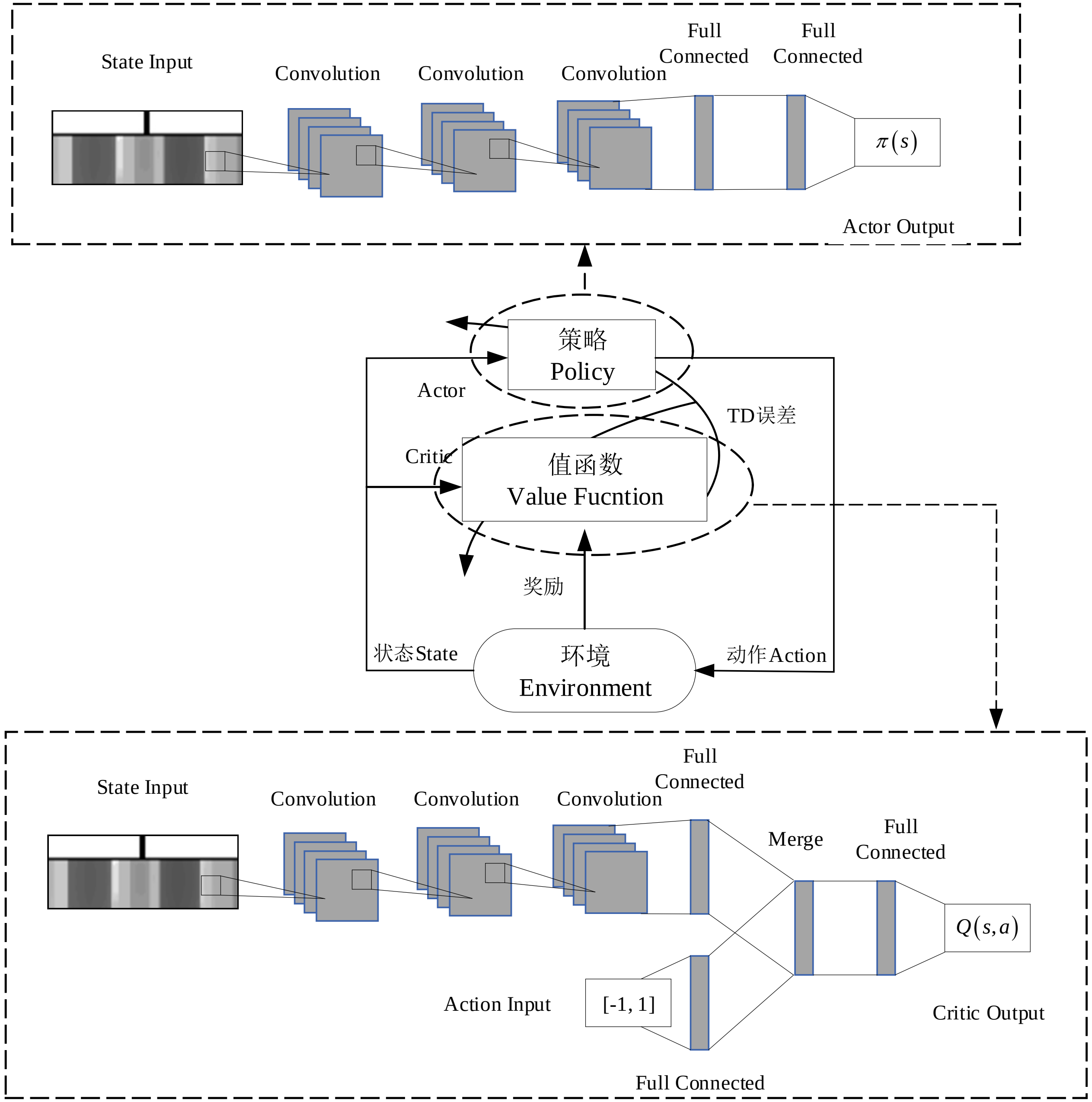
- **DDPG Agent**
- Network Architecture

Actor network

With image input and action output

Critic network

Pre-process state input with CNN
Pre-process action input with FN
Concatenate two pre-process information
and output the policy



4. Results

- **DDPG Agent**
- Hyperparameters and Platform

Platform	
Ubuntu16.04 + Python 3.5 + CUDA + CuDnn + GTX1080TI	
Parameter	**Value**
Training Steps	10,000
Memory Capacity	2,000
Batch Size	32
Discount Factor	0.9
Learning Rate	$[10^{-4},10^{-3}]$
Exploitation probability	1 to 0.1
OU Process Mean	0
OU Process Variance	0.5
OU Process inertia	0.15
Soft Target Update Factor	10^{-2}

Critic Network	Output Shape	Parameter.No
State Input	(None, 1, 30, 100)	
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

Actor Network	Output Shape	Parameter.No
State Input	(None, 1, 30, 100)	
Con2d_1	(None, 32, 7, 25)	544
Con2d_2	(None, 15, 3, 64)	14,464
Con2d_3	(None, 15, 3, 64)	4,160
Dense_1	(None, 512)	1,475,072
Action Input	(None, 1)	
Dense_2	(None, 512)	1024
Merge_1	(None, 512)	0
Dense_3	(None, 512)	262,656
Dense_4	(None, 1)	513
Total		1,758,433

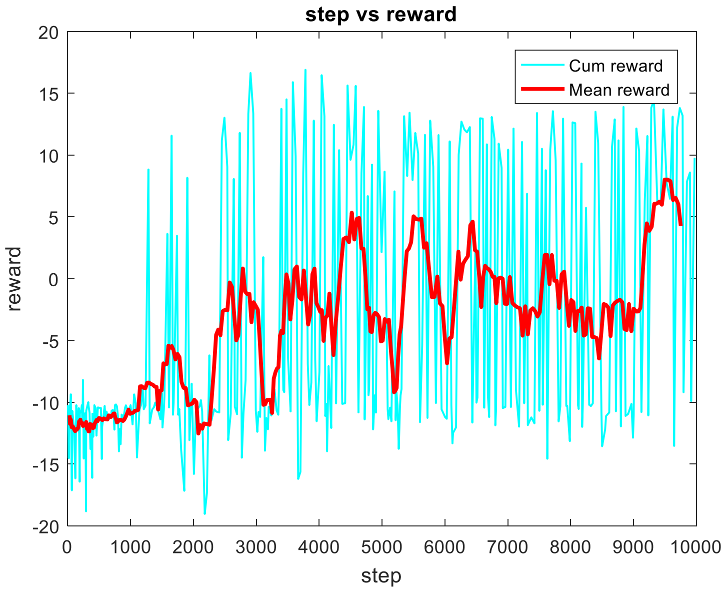
4. Results

- **DDPG Agent**

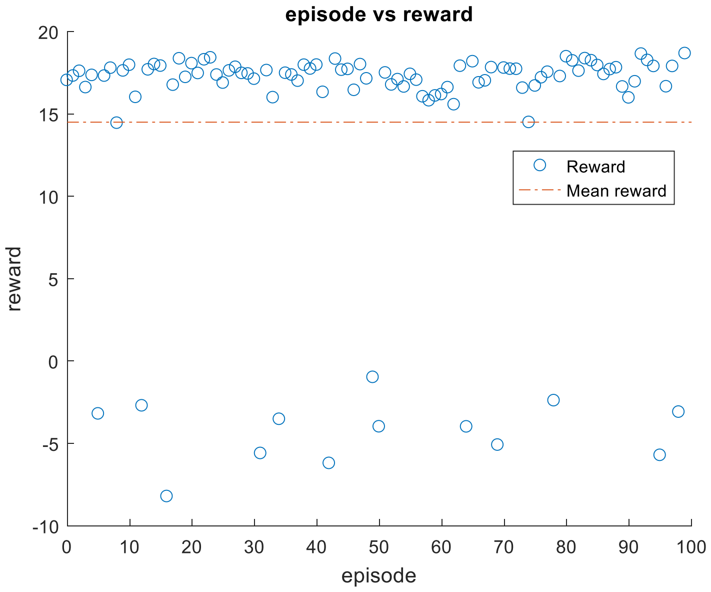
- Video

- **DDPG Agent**

- Results
  - The cumulative reward tend to be relatively stable
  - Evaluation: success: 87%,   average reward: 14.5



Training process



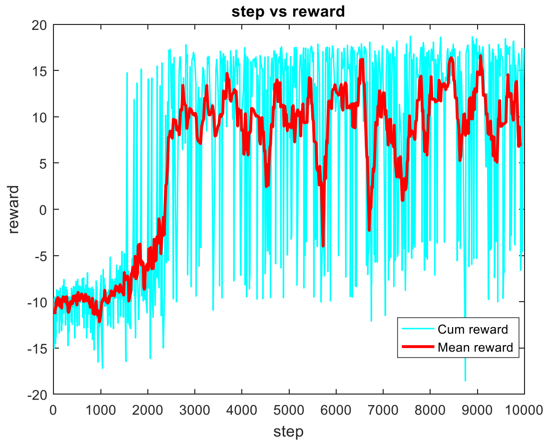
Evaluation

- **Comparison**

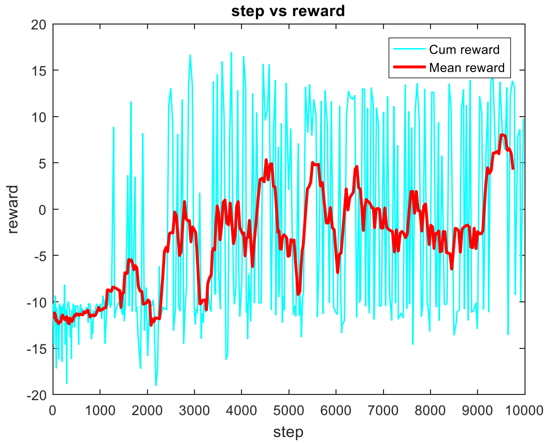
Training Process

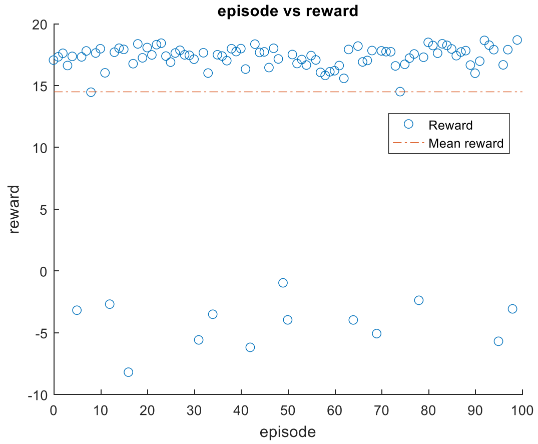
Evaluation

DQN

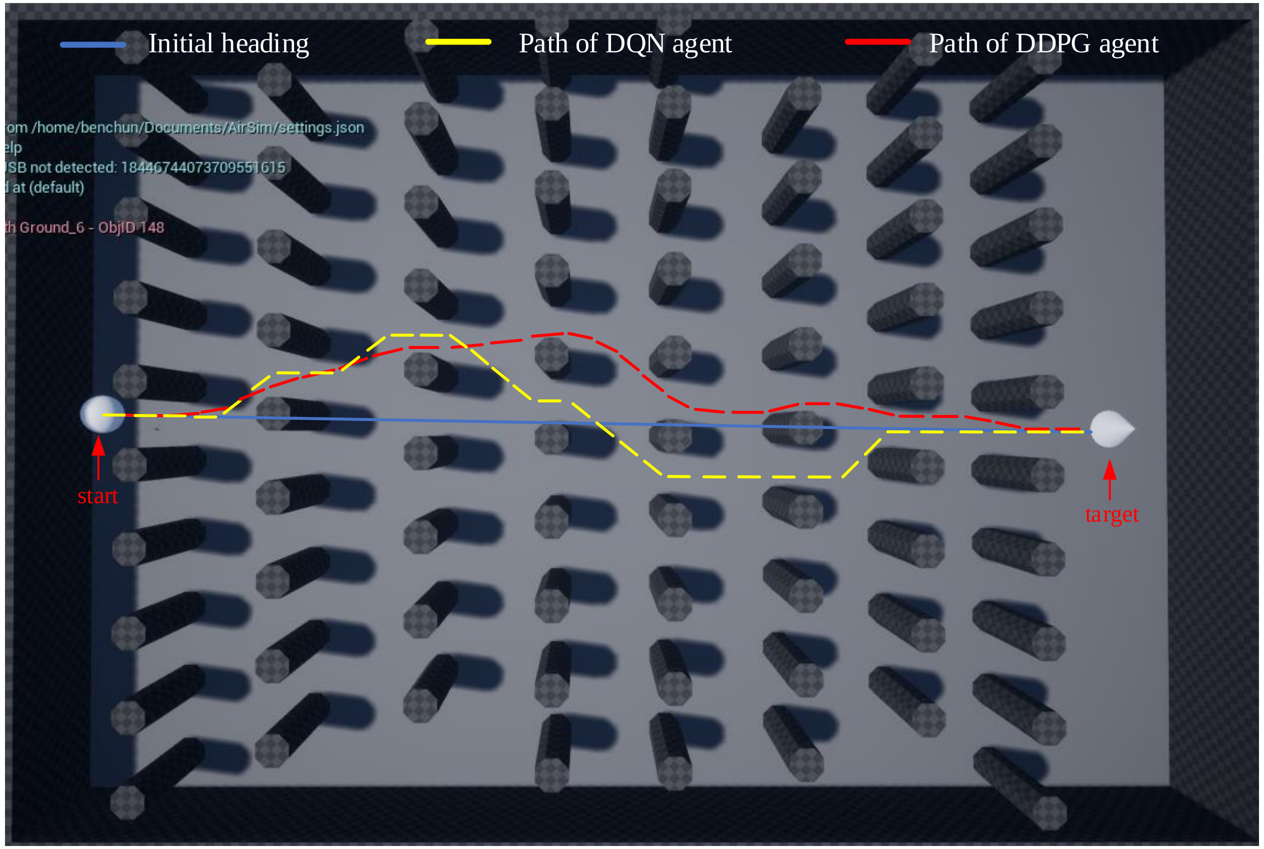


DDPG





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



CONTENTS

5. Discussion

- **Advantages && Disadvantages**

Advantage

Provide a solution in autonomous navigation without building a map

Disadvantage

1. State representation: insufficient
2. Reward: hard to define
3. Training time: too long

- **Future Work**

1. Extensions to multi-UAV system

2. Taking safe navigation into account

3. Transfer to real world

• Conclusion

We applied deep reinforcement learning on autonomous navigation of UAV within a 3D simulated environment.

- Modeled the navigation task as reinforcement learning problem in simulator
- Employed DQN algorithm to complete the navigation task
- Implemented DDPG algorithm to enable UAV act in continuous action space
- Demonstrated the validation of these approaches and make comparison

CONTENTS

6. Future

- **Research Interests**

- Autonomous System
- Robotics
- UAV Navigation

Benchun Zhou

Beihang University, China

15.02.1994 in China | Beijing, China, 100191
(+86)15652388196 | benchun123@buaa.edu.cn

RESEARCH INTERESTS

Robotics, Autonomous Driving, Control Engineering

EDUCATION BACKGROUND

09/2016 - Present — **Beihang University (BUAA)**, Beijing, China (Project 985)
- **M.S., Control Engineering ||| Expected Graduation: 01/2019**
- **Average Score: 92.45/100, Ranking: 2/53**
- Main Courses: Robotic Control System, Path Planning, Reinforcement Learning

01/2018 – 02/2018 — **Ural Federal University (UrFU)**, Yekaterinburg, Russia
- Exchange Experience, Mechanical Engineering
- Average Score: 91/100
- Main Courses: Design of Passenger Cars, Production Engineering

09/2012 – 06/2016 — **Chongqing University (CQU)**, Chongqing, China (Project 985)
- **B.S., Automation**
- **GPA: 3.55/4.0, Ranking 10/203**
- Main Courses: Automatic Control Theory, Embedded System, Optimal Control

INTERNSHIP EXPERIENCE

08/2017 - 02/2018 — **Autonomous Driving R&D Intern ||| Intel China Research Center Ltd. (ICRC)**, Beijing
- Created our own mobile-based robot, mounted some sensors and actuators, and designed base controllers with Arduino.
- Computed and published the laser and odometry.
- Executed Simultaneous Location and Mapping (SLAM) on Robotics Operating System (ROS), built a map from the environment
- Configured and launched navigation stack on the robot with A* and DWA.
- Sent goal to robot using Python and realized autonomous navigation and obstacle avoidance in a known map

RESEARCH EXPERIENCE

09/2017-Present — **Master Thesis ||| Autonomous UAV Navigation Using Deep Reinforcement Learning**
- Investigated the deep reinforcement learning methods on the vision-based path planning of an autonomous drone within a 3D simulated environment.
- Employed Deep Q-network (DQN) algorithm, which learned the values of actions and then selected actions based on their estimated action values.
- Presented a deterministic policy based actor-critic learning framework (DDPG) for UAV path planning in the continuous action space.
- Implemented the reinforcement task on AirSim concerning obstacle avoidance and goal reaching.
- Demonstrated the validation of this approaches yielding high success rate with relatively few steps. (DQN: 94%, DDPG: 87%).
- Submitted a paper to *Journal of Intelligent & Robotic System (SCI)*.

03/2017 – 03/2018 — **Engineering Project ||| High Precision Servo System**
- Modeled the system as a second-order transfer function through the frequency-domain analysis (Bode plot).
- Designed control algorithm (PID + DOB + ZPETC) to overcome the disturbance caused by friction and meet the dynamic requirement.
- Debugged the software on real time system (XP + RTX) using MFC/C++.

09/2015 – 05/2016 — **Competition ||| Freescale Cup National Undergraduate Smart Car Competition**
- Completed a four-wheel car engineering production to follow the track with alternating current (20KHz, 10mA).
- Designed electromagnetic sensor circuit and integrated electrical control system to ensure proper maneuverability of the car.
- Developed and debugged program in collaboration with peers which executed Segment PID in controlling the speed and direction. (velocity = 1m/s)

SELECT PUBLICATION

1. **Benchun Zhou**, Weihong Wang, et.al., "**Autonomous Navigation of UAV with Continuous Action Space.**" [J]. *Journal of Intelligent & Robotic Systems*, 2018 (**SCI** under review)
2. **Benchun Zhou**, Weihong Wang, et.al., "Neural Q Learning Algorithm based UAV Obstacle Avoidance." [C] *the 29th Guidance, Navigation and Control Conference (GNCC2018). IEEE*, 2018 (EI Accept)
3. **Benchun Zhou**, Weihong Wang, "An Improved Nonsingular Fast Terminal Sliding Mode Guidance Law with Impact Angle Constraints." [C]. *the 8th International Conference on Intelligent Control and Information Processing. (ICICIP2017) IEEE*, 2017. (EI Accept)

HONORS AND AWARDS

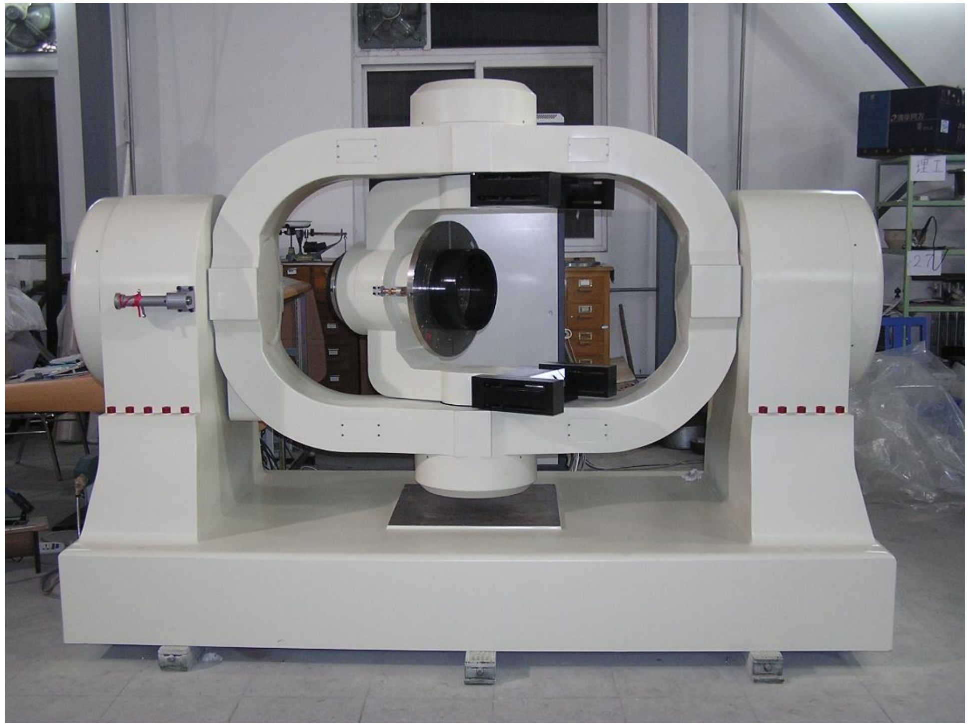
The First Prize Scholarship in Beihang University (Top 3%)	2017
Outstanding Student in Beihang University	2017
National Scholarship (Top 2%)	2015
National Scholarship for Encouragement (Top 5%)	2014
The First Prize Scholarship in Chongqing University (Top 3%)	2014
Outstanding Student in Chongqing University	2014

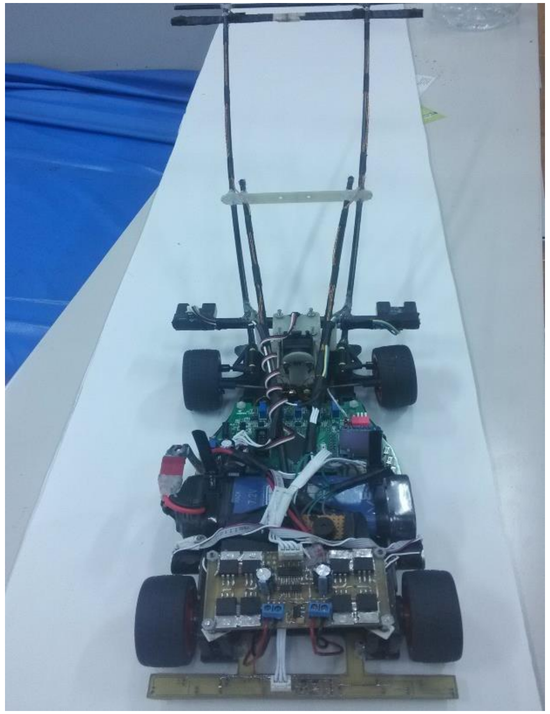
PERSONAL SKILLS

【Language Test Score】	**IELTS: 6.5** (L:7.0, R:8.5, W:5.5, S:5.5)
【Programming Skill】	C/C++, Matlab/Simulink, Python, ROS
【Reinforcement Learning】	Familiar with basic algorithm in DRL, such as DQN, DDPG.
【Path Planning】	Familiar with mainstream path planning methods, such as A*, RRT, DWA, etc.
【Control Algorithm】	Able to analyze the stability of the system in time domain and frequency domain. Able to implement control algorithm, such as PID, LQR, etc.
【Embedded System】	Skilled in hardware design tools, such as AD, Eagle. Skilled in embedded system (STM32/Arduino), with 2 years of practical experience

6. Future

- **Project experience**





6. Future

- **Career Plan**

- **Germany**
  - Global leader in science and technology
  - Prosperous industrialism
  - Culture and openness

THANKS YOU!