Sample Title for HKUST Thesis

by

Ming XIAO

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in the Department of Electronic and Computer Engineering

August 2019, Hong Kong

Authorization

Ming XIAO

August 2019

Sample Title for HKUST Thesis

by

Ming XIAO

This is to certify that I have examined the above MPhil thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Lei LI, Thesis Supervisor

Prof. Meimei Han, Head of Department

Thesis Examination Committee

1. Prof. Lei LI (Supervisor)	Department of Electronic and Computer Engineering
2. Prof. xxx	Department of Electronic and Computer Engineering
3. Prof. xxx	Department of Electronic and Computer Engineering
4. Prof. xxx	Department of Mathematics
5. Prof. xxx (External Examiner)	Department of Electrical Engineering
	Vienna University of Technology

Department of Electronic and Computer Engineering
August 2019

Acknowledgments

First of all, I am truly grateful for being one of the first PhD students supervised by Prof. Li. He was full of passion and patience when helping me build the know-how for this degree. It has been a great pleasure for me to be part of this team and grow together with the lab during the last four years. Furthermore, I would like to thank all of the members of the thesis examination committee for their careful examination of my thesis.

Finally, I would not stand at this current point without their endless love and unconditional support for all these years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Sample Title for HKUST Thesis

by Ming XIAO

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

Abstract

Every copy of the thesis must have an English abstract. Abstracts must provide a concise summary of the thesis in **300** words or less.

Theses should be written in English. Students in the School of Humanities and Social Science who are pursuing research work in the areas of Chinese Studies, and who can demonstrate a need to use Chinese to write their theses should seek prior approval from the School via their thesis supervisor and the divisional head. If approval is granted, students are also required to produce a translation of the title page, authorization page, signature page, table of contents and the abstract of thesis in English.

(1)A thesis should be presented in a permanent and legible form either in original typescript, or a comparably permanent process. (2)Paper and print quality are vitally important for legibility. Normally, international size A4 (297mm x 210mm) 80 to 90 gsm woodfree white paper of good quality should be used for the thesis. Exceptionally, paper other than international A4 size may be used when the nature of the thesis requires it. Any thesis not typed or printed on the correct size of paper will not be accepted. (3)The final thesis copies should be printed on both sides of the paper. (4)A conventional font, size 12-point, 10 to 12 characters per inch must be used. One-and-a-half line spacing should be used throughout the thesis, except for abstracts, indented quotations or footnotes where single line spacing may be used. (4)All margins should be consistently 25mm (or at a maximum of 30mm) in width. The same margins should be used throughout a thesis. Exceptionally, margins of a different size may be used when the nature of the thesis requires it.

CHAPTER 1

INTRODUCTION

Intelligent robots are influencing many aspects of the world nowadays, from collaborative robot arms in factories to L4 autonomous driving technology, from biped household robots to quadruped mobile military agents, and from unmanned surface vehicles to quadrotor swarms. The more deeply we imagine the future, the more indispensable we find robot services.

For mobile robots, navigation is always the kernel function. However, compared with the automation of manipulation, mobile robot navigation is evolving more slowly. In the classical manipulation and manufacturing scenarios, the workspace of robots is usually well defined, and the robots can perform correctly under human-designed programming, without any interaction between uncertain objects and themselves. If we regard *replacing the repetitive workloads* as the first step, we should start to consider *living with the robot* in the next step. For mobile robots, to be incorporated into human's daily life, they must be intelligent in unknown and pedestrian-rich environments for tasks like autonomous driving, cargo delivers and household mobile robots, which is *behaving like a human.*

Human beings can navigate in crowded environments smartly without dependence on pre-defined high-resolution maps. We can also explore unknown environments without taking time to think about the information gain or frontiers. To navigate safely and efficiently like a human being, mobile robots should be able to perceive and predict behaviours of unfamiliar and dynamic agents. Based on the implicit or explicit understanding, an accessible policy considering various constraints is needed to guide the robots.

Deep learning, as a solution for artificial intelligence, is capable of building progressively meaningful feature abstraction of input data. It plays an essential role in various fields of study [16], bringing the state of the art in image classification [19, 20, 23], semantic segmentation [6, 31], human-level game playing [33, 35], driving real robotic systems in navigation [47, 54, 55] and manipulation [26, 51] tasks.

We may be witnessing the most rapidly growing trend of deep learning techniques for robotics tasks in recent years. Replacing hand-crafted features with learned hierarchical distributed deep features, and learning control policies directly from high-dimensional sensory inputs, the robotics community is making substantial progress towards building fully autonomous intelligent systems.

Part I

From Supervisd Learning to Reinforcment Learning

CHAPTER 2

ACTIVE OBSTACLE AVOIDANCE LEARNING

2.1 Introduction

2.1.1 Motivation

Obstacle avoidance is fundamental for mobile robots in divers tasks, like cleaning, mining, and rescue, etc. In this chapter, we deal with a classic task for a mobile robot equipped with a depth sensor: attempting to avoid collisions with obstacles while navigating in an unknown environment. Using stereo vision systems or radar sensors as aids, researchers often build the geometry or topological mapping of environments [29, 30] to make navigation decisions based on a global representation. Such methods regard the environment as a geometrical world and decisions are only made with preliminary features without an online learning process. Specific logic has to be particularly designed for different environments. It is still a challenge for mobile robots to rapidly adapt to a new environment. Active learning for mobile robots to achieve tasks have been widely achieved for mobile robot navigation tasks like localisation [3, 4]. In this chapter, we aim for robots to actively learn essential features for obstacle avoidance.

Convolutional neural network (CNN), a typical model for deep-learning and cognitive recognition, are the state-of-the-art in computer vision tasks. The Success of type of hierarchical model also motivates robotics scientists to apply deep learning algorithms in common robotics problems like recognition and obstacle avoidance [15, 36, 43].

As with most supervised learning algorithms, CNN extracts feature representations through training with a massive amount of labelled samples. However, unlike typical computer vision tasks, robot navigation usually happens in dynamic environments with high probability and uncertainty. The overfitting problem of supervised learning limits the perception ability of hierarchical models for unseen inputs, and it is unrealistic for mobile robots to collect datasets covering all of the possible conditions. Besides this, the time-consuming work of dataset collection and labelling seriously influences the application of CNN-based learning methods. Another common problem is that robotics research usually considers the CNN mechanism as a black box. There lacks a proper metric to validate the efficiency of the network, let alone the improvement. In this chapter, we use receptive fields to visualize the salient regions that determine the output. This provides the ground for structure selection and performance justification.

Reinforcement learning is an efficient way to learn control policies without referencing the ground-truth. Combining reinforcement learning and hierarchical sensory processing, deep reinforcement learning (DRL) [35] can learn optimal policies directly from high-dimensional sensory inputs. And it has been shown to outperform all of the previous artificial control algorithms on Atari games [34].

We have proved the feasibility of a CNN-based supervised learning method for obstacle avoidance in an indoor environment [43, 44] and the effectiveness of the conventional reinforcement learning method in policy estimation [45, 46] through feature representations extracted from the pre-trained CNN model in [43]. In this chapter, we propose an end-to-end deep reinforcement learning method towards active obstacle avoidance in an unfamiliar environment by taking depth images as the input and control commands as the output. Unlike conventional learning methods, the training of deep reinforcement learning is a cognitive process.

2.1.2 Contribution

We stress the following contributions and features of this chapter:

- By deep reinforcement learning, we show the developed obstacle avoidance capability of a mobile robot in unknown environments. We initialize the weights from a previous CNN model trained with real-world sensory samples and continually train it in an end-to-end manner. The performance is evaluated in both simulated and real-world environments.

- The deep reinforcement learning model can quickly achieve obstacle avoidance ability in an indoor environment with several thousand training iterations, without additional man-made collection or labelling work for datasets.

- For evaluations of the CNN, we use receptive fields in the original inputs to reason the feasibility of the trained model. The receptive fields activated by the final feature representations are presented through bilinear upsampling. The activation characters prove the cognitive ability improvement of hierarchical convolutional structures for traversability estimation.

2.2 Related Work

Conventional obstacle avoidance strategies mainly depend on hand-crafted features extracted from environments [24]. Benefiting from the development of large-scale computing hardware like GPUs, deep learning related methods have been considered to address robotics problems, including obstacle avoidance.

2.2.1 Deep Learning in Robotics Obstacle Avoidance

CNN has been applied to recognize off-road obstacles [36] by taking stereo images as input. It also helps aerial robots to navigate along forest trails with a single monocular camera [15]. In [43], a three-layer convolutional framework was used to perceive an indoor corridor environment for mobile robots. By taking raw images or depth images as inputs, and taking the moving commands or steering angles as outputs, the weights of CNN-based models could be trained through back-propagation and stochastic gradient descent.

Note that the supervised learning methods mentioned above require a large amount of effort for collecting and labelling of datasets. Kim *et al.* [22] achieved the labelling result by using other sensors with higher resolution. Tao *et al.* [48] labelled the centre sample of the clustering result for object classification as a semi-supervised method. Considering the requirement for auxiliary judgments, unsupervised learning methods do not eliminate the labelling work essentially.

Deep learning in raw image processing has shown significant potential to solve visual-based robot control problems. However, even though CNN related methods have accomplished many breakthroughs and challenging benchmarks for vision perception tasks like object detection and image recognition, applications in robotics control are still limited.

2.2.2 Reinforcement Learning in Robotics

Reinforcement learning is a useful way for robots to learn control policies. The main advantage of reinforcement learning is the completed independence from human-labelling.

Motivated by the trial-and-error interaction with the environment, the estimation of the action-value function is self-driven by taking the robot states as the input of the model. Conventional reinforcement learning methods improved the controller performances in path-planning of robot-arms [50] and control of helicopters [38].

Through regarding RGB or RGB-D images as the states of robots, reinforcement learning can be directly used to achieve visual control policies. In our previous work [46], a Q-learning based reinforcement learning controller was used to help a *Turtlebot* navigate in the simulation environment.

2.2.3 Deep Reinforcement Learning

Due to the potential of automating the design of data representations, deep reinforcement learning has attracted considerable attention recently [10]. Deep reinforcement learning was firstly applied on playing 2600 Atari games [35], where the typical model-free Q-learning method was combined with convolutional feature extraction structures as a Deep Q-network (DQN). The learned policies beat human players and previous algorithms in most of the Atari games. Based on the success of DQN [35], revised deep reinforcement learning methods appeared to improve the performance for various applications. Different to DQN, which takes three continuous images as input, DRQN [18] replaces several regular convolutional layers with recurrent neural network (RNN) and long short term memory (LSTM) layers. Taking only one frame as the input, the trained model performed as well as DQN in Atari games. Duelling network [49] separated the Q-value estimator into two independent network structures, one for the state value function and one for the advantage function. Now, it is the state-of-art method on the Atari 2600 domain.

For robotics control, deep reinforcement learning has also accomplished various simulated robotics control tasks [27]. In the continuous control domain [17], the same model-free algorithm robustly solved more than 20 simulated physics tasks. Control policies are learned directly from raw pixel inputs. Considering the complexity of control problems, a model-based reinforcement learning algorithm was proved to be able to accelerate the learning procedure in [27] so that the deep reinforcement learning framework could handle more challenging problems.

No matter whether in Atari games or the control tasks mentioned above, deep reinforcement learning has been showing its advantages in simulated environments. However, it is rarely used to address robotics problems in real-world environments. As in [53], the motion control of a *Baxter* robot motivated by deep reinforcement learning could make sense only with simulated semantic images but not raw images taken by real cameras. Thus, we consider the feasibility of deep reinforcement learning in real-world tasks to be the primary contribution of our work.

2.3 Implementation of Deep Reinforcement Learning

One of the main limitations of applying deep reinforcement learning in real-world environments is that the repeated trial-and-error learning procedure of reinforcement learning is quite difficult to implement in the actual world. In Atari games, the controller must repeat the games thousands of times after each attempt. However, in the physical world, it is unrealistic for robots to repeat the same task with the same beginning state again and again. In this section, we implement the same model-free deep reinforcement learning framework as DQN [35] in the simulation environment. The weights of CNN are initialised from a supervised learning model with data collected from actual environments. In the end-to-end training procedure, we set a small learning rate for the gradient descent of the data representation structure compared with the learning rate used in the training of the supervised learning model [43]. Finally, the learned model can both keep the navigation ability in the original world and build the adaptation for an unknown world.

2.3.1 Simulated Environment

In our previous work [43], the training datasets were collected in structured corridor environments. Depth images in these datasets were labelled with real-time moving commands from human decisions. In this chapter, to extend the navigation ability of the mobile robot, we set up a more complex indoor environment, as shown in Fig. 2.1, in the *Gazebo*[1] simulator. Besides the corridor-like traversable areas, there are much more complicated scenes like cylinders, sharp edges and multiple obstacles with different perceptive depths. These newly created scenes have never been used in the training of our previous supervised learning model [43].

We use a *Turtlebot* as the main agent in the simulated environment. A *Kinect* RGBD

[1]http://gazebosim.org

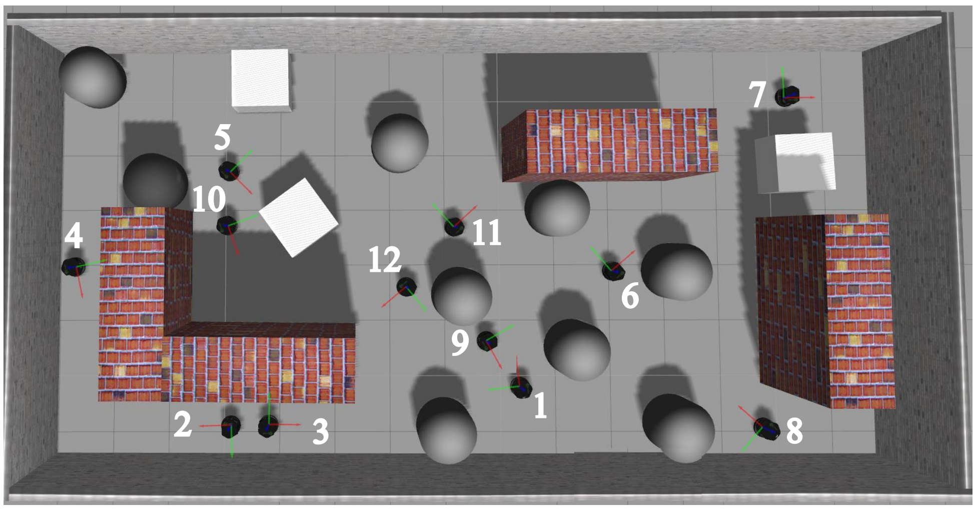


Figure 2.1: The simulated indoor environment implemented in *Gazebo* with various scenes and obstacles. The *Turtlebot* is the experimental agent with a *Kinect* camera mounted on it. One of the 12 locations marked in the figure is randomly set as the start point in each training episode. The red arrow of every location represents the initial moving direction.

camera is mounted on top of the robot. We can receive the real-time RGB-D raw image from the field of view (FOV) of the robot. All of the requested information and communications between agents are achieved through *ROS*[2] interfaces.

2.3.2 Deep Reinforcement Learning Implementation

As a standard reinforcement learning structure, we set the environment mentioned in Section 2.3.1 as e. At each discrete time step, the agent selects an action a_t from the defined action set. In this section, the action set consists of five moving commands, namely *left, half-left, straight, half-right* and *right*. Detailed assignments of speeds related to the moving commands are introduced in Section 2.4. The only perception by the robot is the depth image x_t taken from the *kinect* camera after every execution of the action. Unlike in Atari games, where the reward r_t is the change of the game's score, the only feedback used as the reward is a binary state, indicating whether a collision occurs or not. This is decided by checking the minimum distance l_t through the depth image taken by the *Kinect* camera. Once a collision occurs, we set a negative reward of t_{ter} to represent the termination. Conversely, we grant a positive reward of t_{move} to encourage collision-free movement.

The state sequences s_t in the simulated environment are regarded as a *Markov Decision Process* (MDP) which is an alternate combination of moving commands and depth-image

[2]http://www.ros.org

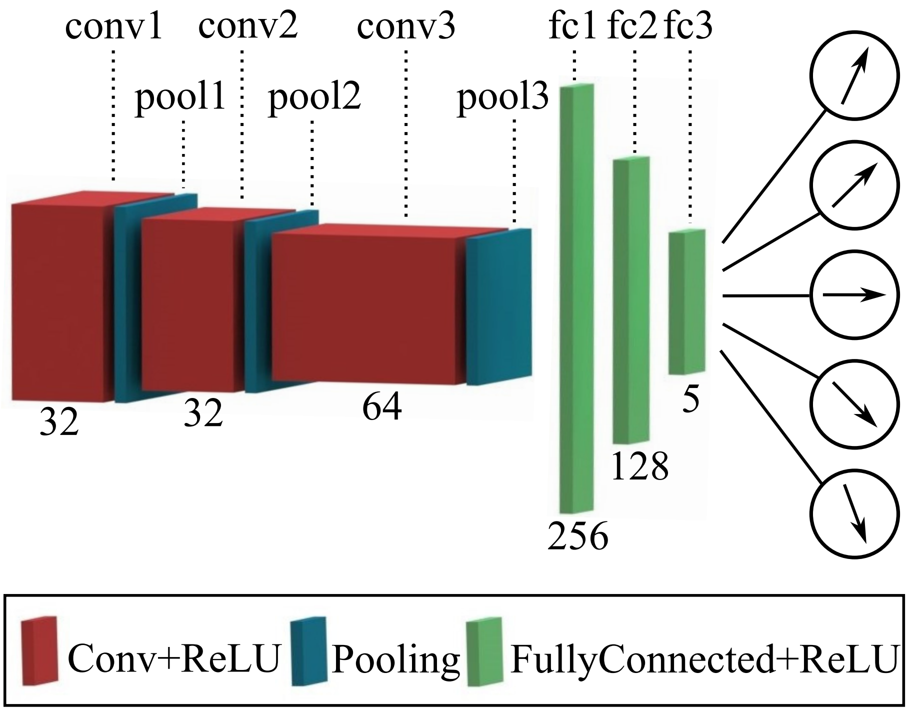


Figure 2.2: The network structure for the actor-evaluation estimation. It is a combination of convolutional networks for feature extraction and fullyconnected layers for policy learning. They have been separately proven to be effective in our previous works [43, 46].

states, where $s_t = \{x_1, a_1, x_2, a_2, \ldots, a_{t-1}, x_t\}$. The sequence terminates once the collision happens. As the assumption of the MDP, x_{t+1} is completely decided by (x_t, a_t) without any reference to the former states or actions in s_t. The sum of the future rewards until the termination is R_t. With a discount factor γ for future rewards, the sum of future estimated rewards is $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T means the termination time-step. The target of reinforcement learning is to find the optimal strategy π for the action decision through maximizing the action-value function $Q^*(x,a) = max_\pi \mathbb{E}[R_t | x_t = x, a_t = a, \pi]$. The essential assumption in DQN [35] is the *Bellman equation*, which transfers the target to maximize the value of $r + \gamma Q^*(x', a')$ as

$$Q^*(x,a) = \mathbb{E}_{x' \sim e}[r + \gamma \max_{a'} Q^*(x', a') | x, a].$$

Here, x' is the state after acting action a in state x. DQN estimates the action-value equation by CNN with weights θ, so that $Q(s, a, \theta) \approx Q^*(s, a)$.

In this section, we use three convolutional layers for feature extractions of the depth image and use an additional three fullyconnected layers for obstacle avoidance learning. The structure is depicted as red and green cubes in Fig. **??**. To increase the non-linearity for better data fitting, each Convolutional or Fullyconnected layer is followed by a Recti-

Algorithm 1 Deep reinforcement learning algorithm

```
1: Initialize the weights of evaluation networks as $\theta^-$
   Initialize the memory $D$ to store experience replay
   Set the collision distance threshold $l_s$
2: for episode $= 1, M$ do
3:   Randomly set the Turtlebot to a start position
     Get the minimum intensity of depth image as $l_t$
4:   while $l_t > l_s$ do
5:     Capture the depth image $x_t$
6:     With probability $\varepsilon$ select a random action $a_t$
       Otherwise select $a_t = \text{argmax}_a Q(x_t, a; \theta^-)$
7:     Move with the selected moving command $a_t$
       Update $l_t$ with new depth information
8:     if $l_t < l_s$ then
9:       $r_t = r_{ter}$
         $x_{t+1} = Null$
10:    else
11:      $r_t = r_{move}$
         Capture the new depth image $x_{t+1}$
12:    end if
13:    Store the transition $(x_t, a_t, r_t, x_{t+1})$ in $D$
       Select a batch of transitions $(x_k, a_k, r_k, x_{k+1})$ randomly from $D$
14:    if $r_k = r_{ter}$ then
15:      $y_k = r_k$
16:    else
17:      $y_k = r_k + \gamma \max_{a'} Q(x_{k+1}, a'; \theta^-)$
18:    end if
       Update $\theta$ through a gradient descent procedure on the batch of $(y_k - Q(\phi_k, a_k; \theta^-))^2$
19:  end while
20: end for
```

fied Linear Unit (ReLU) activation function layer. The number under each Conv+ReLU or FullyConnected+ReLU cube is the number of channels of the output data related to this cube. The network takes one depth raw image as the input. The five channels of the final fully-connected layer *fc3* are the values of the five moving commands. Also, to avoid the overfitting in the training procedure, both of the first two fully-connected layers *fc1* and *fc2* are followed with a dropout layer. Note that dropout layers are eliminated in the test procedure [42].

Algorithm 1 shows the workflow of our revised deep reinforcement learning process. Similar to [35], we use the memory replay method and the ε-greedy training strategy to control the dynamic distribution of training samples. After the initialization of the weights for the convolutional networks shown in Fig. **??**, we set a distance threshold l_s to check if the *Turtlebot* collides with any obstacles. At the beginning of every episode, the *Turtlebot* is randomly set to a start point among the 12 pre-defined start points shown in Fig. 2.1. This extends the randomization of the *Turtlebot* locations from the whole simulation world and keeps the diversity of the data distribution saved in memory replay for training.

For the update of weights θ, y_k is the target for the evaluation network. It is calculated by summing the instant reward and the future expectation estimated by the networks with the former weights, as mentioned before in the *Bellman equation.* If the sampled transition is a collision sample, the evaluation for this (x_k, a_k) pair is directly set as the termination reward r_{ter}. Setting the training batch size to be n, the loss function is

$$L(\theta_i) = \frac{1}{n}\sum_{k}^{n}[(y_k - Q(x_k, a_k; \theta_i))^2].$$

After the estimation of $Q(x_k, a_k)$ and $\max_{a'} Q(x_{k+1}, a')$ with the former θ^-, the weights θ of the network will be updated through back-propagation and stochastic gradient descent.

2.4 Experiment

2.4.1 Training

At the beginning of the training, convolutional layers are initialized by copying the weights trained in [43] for the same layer structure. A simple policy learning networks structure was also separately proved in [46] with three moving commands as output.

Table 2.1: Training parameters and the related values.

Parameter	Value
batch size	32
replay memory size	3000
discount factor γ	0.85
learning rate	0.0000001
gradient momentum	0.99
distance threshold l_s	0.55
negative reward t_{ter}	-100
positive reward t_{move}	1

Compared with the step-decreasing learning rate in the training of the supervised learning model [43], here we use a much smaller fixed learning rate in the end-to-end training for the deep reinforcement learning model. As the only feedback to motivate the network convergence, the negative reward for the collision between the robot and obstacles must be very large, as in [46]. The training parameters are shown in Table 2.1 in detail. All models are trained and tested with Caffe [21] on a single NVIDIA GeForce GTX 690.

Table 2.2: Speeds of different moving commands.

	angular velocity (rad/s)					line velocity
	Left	H-Left	Straight	H-right	Right	(m/s)
Train	1.4	0.7	0	-0.7	-1.4	0.32
Test	1.2	0.6	0	-0.6	-1.2	0.25

Table 2.3: The average counts of moving steps and the average moving distances in every start point.

	Model	1	2	3	4	5	6	7	8	9	10	11	12
Count	SL	7.6	16.4	33.4	3.0	21.1	20.2	6.9	14.0	16.7	5.6	10.6	19.6
	RL	16.7	**102**	4.0	19.1	14.5	7.4	5.0	3.0	16.8	7.0	18.4	36.6
	DRL 500	6.6	3.7	5.8	4.0	5.0	3.5	4.0	9.8	8.3	4.0	21.0	36.3
	DRL 4000	16.2	13.7	28.7	26.2	26.2	**26.9**	10.4	41.8	13.0	20.3	12.4	23.1
	DRL 7500	44.3	38.0	16.4	31.1	23.4	23.2	**29.1**	30.8	18.1	24.6	27.7	21.4
	DRL 40000	**136**	71.9	**66.1**	**152**	**91.7**	17.8	14.3	**103**	**158**	**86.0**	**102**	**113**
Dist.	SL	1.1	2.1	7.3	0.2	3.8	6.1	0.8	2.6	2.6	0.5	1.6	3.2
	RL	3.6	3.0	0.2	5.4	1.8	0.7	0.2	0.2	4.1	0.6	3.9	9.0
	DRL 500	1.1	0.5	0.5	0.6	0.7	0.5	0.5	0.7	0.8	0.6	0.8	0.9
	DRL 4000	7.5	4.4	**17.6**	10.3	**15.8**	**10.7**	2.1	20.3	2.4	5.7	2.8	10.1
	DRL 7500	11.0	**20.5**	6.0	10.6	9.4	9.0	**15.3**	12.8	4.1	8.3	5.5	7.4
	DRL 40000	**39.5**	18.6	14.3	**21.0**	7.6	10.5	2.7	**50.2**	**33.0**	**67.1**	**19.2**	**39.9**

Table 2.2 lists the assignments of speeds for the five output moving commands both in training and testing procedures. All of the training or testing commands have the same line velocity. The various moving directions are declared with different angular velocities. The speeds for the training procedure are a little larger than the speeds for testing. With a higher training speed, the robot is motivated to collide aggressively and there will be more samples with negative rewards in the replay memory. In the testing procedure, a small speed can keep the robot making decisions more frequently.

Fig. 2.3 presents the loss reduction along training iterations. At each iteration, a batch including 32 depth images is randomly chosen from the replay memory. Unlike the training of conventional supervised learning methods, the loss of deep reinforcement learning may not converge to zero, depending extremely on the declaration of the negative

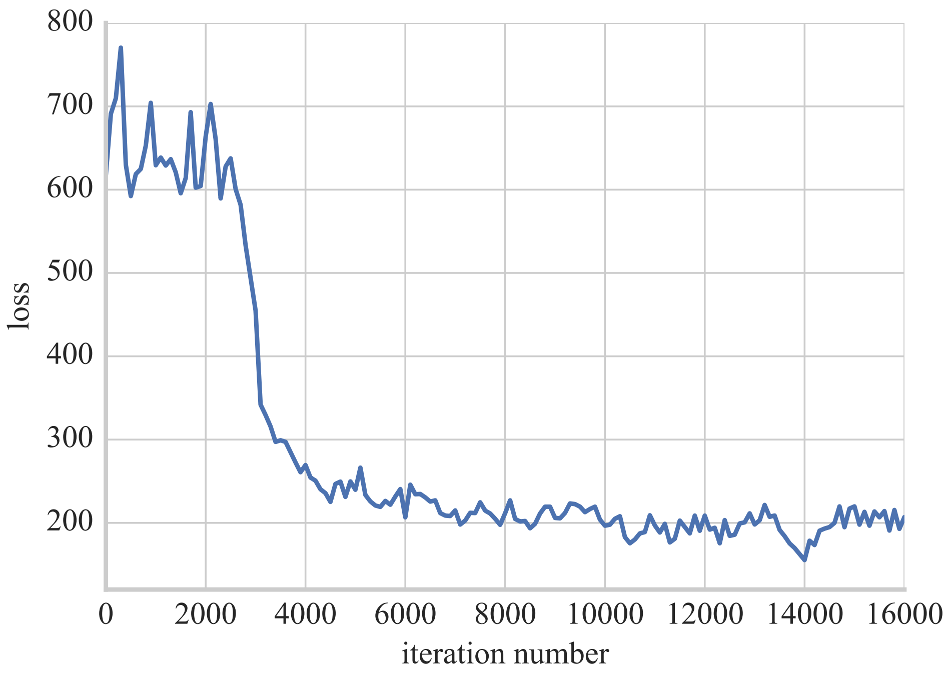
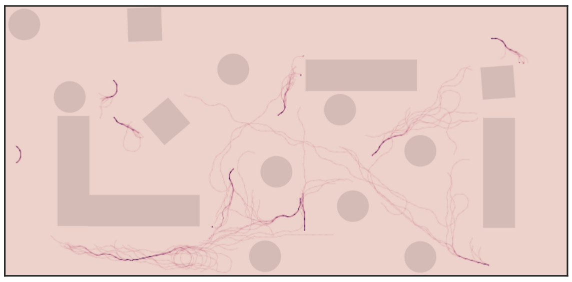


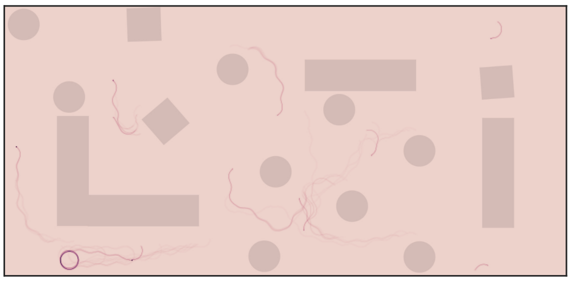
Figure 2.3: The loss decreasing curve as the training iterates. There is a batch of 32 samples used to do back-propagation in every iteration.

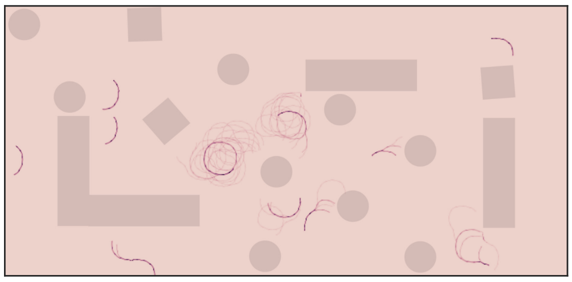
reward. Among the estimation Q-values for state-action pairs, the maximum represents the optimal action. The value itself can limited present the sum of the future gains [35]. As seen in the figure, the loss converges after 4000 iterations. Test results of several trained models after 4000 iterations are compared in Section 2.4.2.

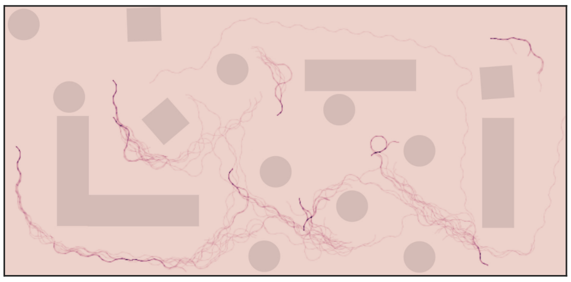
2.4.2 Analysis of Policy Tests

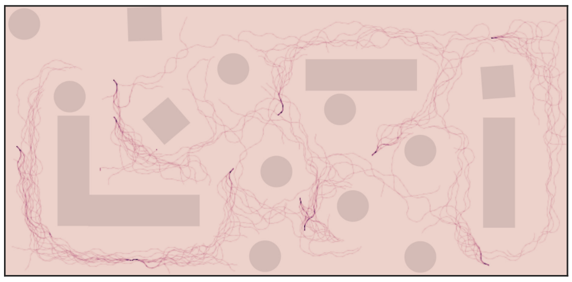
We firstly look at the obstacle avoidance capability of the trained model. The trained DRL models after 500, 4000, 7500, and 40000 iterations are chosen to be tested in the simulated environment. The trained supervised learning (SL) model from [43] and the reinforcement learning (RL) model from [46] are compared directly, without any revision of the model structure or tuning for the weights. In all 12 start points shown in Fig. 2.1, every model starts ten episodes with the test speeds listed in Table 2.2 for the five moving commands. Additionally, every test episode will stop automatically after 200 moving steps, so that the robot will not explore freely forever. With the same CNN structure for all the trained models, the forward prediction takes $48(\pm 5)ms$ for each raw depth input. After the forward calculation of the received real-time depth image, the robot chooses the moving command with the highest evaluation. The average counts of the moving steps for each start point are listed in Table 2.3. The more moving steps, the longer the time the robot has been freely moving in the simulated environment without collision.


(a) Supervised Learning


(b) Reinforcment Learning


(c) DRL 500 iterations


(d) DRL 4000 iterations


(e) DRL 7500 iterations

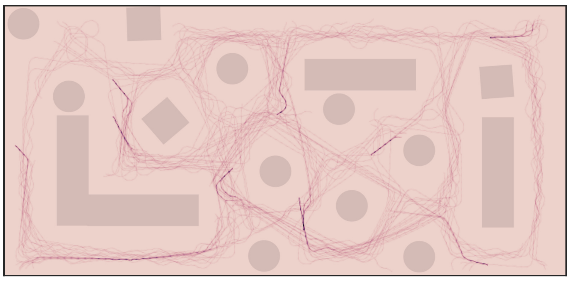

(f) DRL 40000 iterations

Figure 2.4: Heatmaps of the trajectory points' locations in the 10 test episodes of each model for all 12 start points. The counts of points in every map grid are normalized to [0,1]. Note that the circles at the left-bottom corner of (b) and the middle of (c) are actually a stack of circular trajectories caused by the actual motion of the robot.

The moving trajectory points of each model starting from all 12 positions are recorded. Fig. 2.4 depicts the trajectory after normalizing the counts of the trajectory points to $[0, 1]$ in each map grid of the training environment. The trained model may choose the *left* or *right* command to rotate the robot in place so that no collisions will happen, as depicted in the circles in the left-bottom corner of Fig. 2.4b and the middle of Fig. 2.4c, for example. The very large moving count of the RL model in column 2 of Table 2.3 corresponds to the trajectory circle in Fig. 2.4b. To avoid the appearance of this local minimum, the distances between the start and the endpoint are recorded as an additional evaluation metric. Note that, after a long exploration time, the robot may move back to the area near the start point. So the distance may not be perfectly equal to the obstacle avoidance ability of the trained model.

From the heat maps in Fig. 2.4, we can see that the SL model cannot be adapted to the simulated environment, especially in the scenes with multiple obstacles at different depths and the RL model gives the worst of the results. In the training of the RL model [46], only the weights of the fully-connected network for policy iterations were updated iteratively. The DRL model shows significant improvement compared with the RL model because its training is end-to-end. Thus, not only is the policy network (*fc* layers in Fig. **??**) developed for complicated scenes, but also the CNN model for feature representations.

Seen in Fig. 2.4(c-f), the training for the obstacle avoidance ability of the DRL model is an online-learning process. In the 500-iteration case, the robot always chooses the same moving direction for any scenes. After 4000 iterations, it can be adapted to parts of the environment. However, In the 7500-iteration case, the robot can almost move freely in the whole simulated world. Furthermore, the robot usually chooses the optimal moving direction after 40000 iterations, like the more efficient trajectories in Fig. 2.4f. Unlike the fixed training datasets of the RL model, newly collected training samples of the DRL model are saved to replay memory increasingly. The evaluations of the training scenes are calculated by the current model, which is updated with the increasing number of training iterations. Thus, the robot obstacle avoidance ability will be increased over time.

The 40000-iteration case can almost thoroughly explore the trained environment. Considering the very long training time(12 hours) for 40000 iterations, we choose the model after 7500 iterations to do further analysis. The training time for 7500 iterations is 2.5 hours. This confirms that the mobile robot can be adapted to an unfamiliar environment by transferring the weights of the pre-trained SL model to the DRL framework with very short-term and end-to-end DRL training.

2.4.3 Analysis of Receptive Fields in the Cognitive Process

CNN models are usually considered to be black boxes and the internal activation mechanism of CNN is rarely analyzed. In [52], the strongest activation areas of the feature representations are presented by backtracking the receptive field in the source input. We propose a backtracking method by multiplying the last layer of feature representations (*pool3* in Fig. **??**) with a single channel convolutional filter. The dimension of *pool3* is $64 \times 20 \times 15$ in this chapter. We multiply it with a convolutional kernel of size $1 \times$

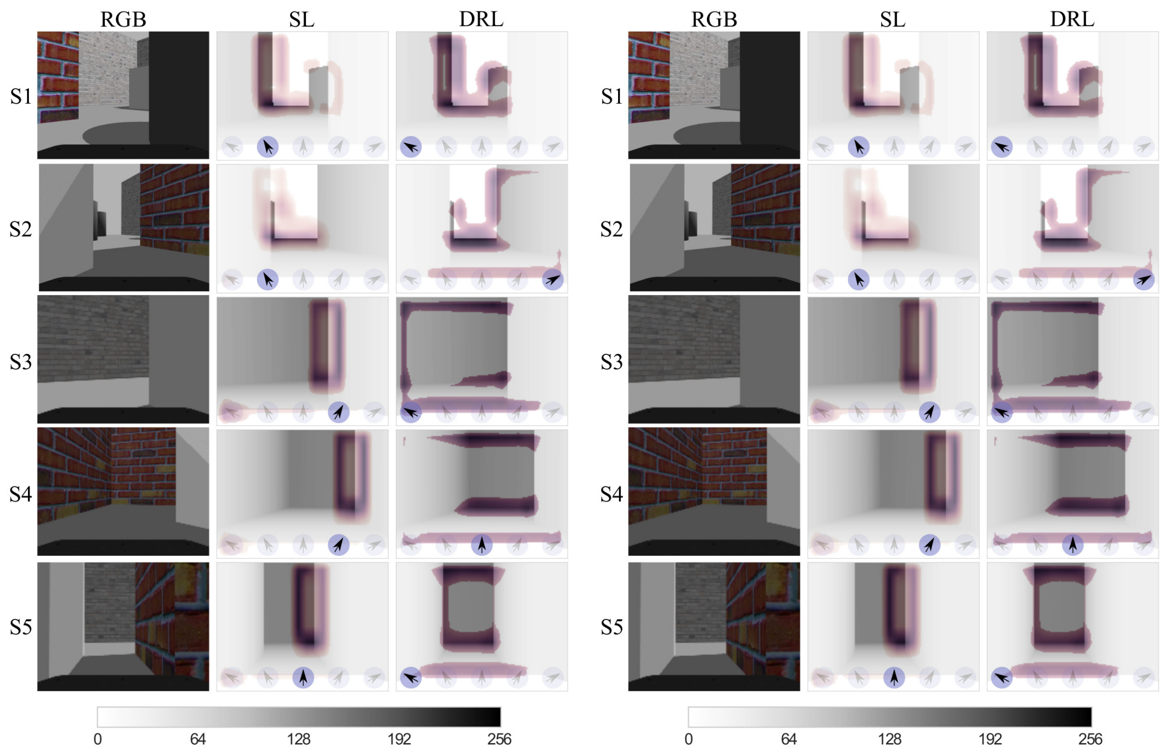
15×15, which is fixed with bilinear weights like the one used for upsampling of semantic segmentation in [31]. After that, a 120×160 matrix of the same size as the input images.

Table 2.4: Evaluations of moving commands for different scenes.

		Left	H-Left	Straight	H-Right	Right
S1	SL	23.9	**26.3**	−10.6	−35.3	−34.4
	DRL	**−16.3**	−36.2	−31.7	−38.7	−44.5
S2	SL	−18.2	**−0.1**	46.2	−30.1	−8.8
	DRL	−30.0	−25.4	−19.8	−21.2	**−15.3**
S3	SL	4.2	5.9	−21.3	**7.7**	−0.2
	DRL	**−5.3**	−15.3	−13.0	−16.5	−19.9
S4	SL	−4.0	−5.5	−15.2	**13.3**	−0.4
	DRL	−4.6	−5.1	**−3.8**	−4.8	−4.3
S5	SL	−17.4	12.3	**23.0**	−9.7	−18.4
	DRL	**−9.9**	−19.2	−15.8	−19.2	−21.6
R1	SL	−38.2	17.5	**27.4**	−15.6	−1.9
	DRL	−84.1	−89.3	−71.8	−78.8	**−63.2**
R2	SL	−11.6	**45.6**	−50.3	4.1	−8.3
	DRL	−8.3	−9.3	**−7.1**	−8.6	−7.5
R3	SL	−18.4	32.2	**37.2**	−23.3	−41.4
	DRL	**−87.7**	−113.3	−90.9	−103.6	−96.3
R4	SL	4.2	1.1	−7.8	−10.1	**15.2**
	DRL	**−87.6**	−141.1	−115.2	−134.8	−137.4
R5	SL	0.4	−6.2	−48.8	**32.8**	11.9
	DRL	−3.4	−3.3	**−2.5**	−3.0	−2.7

We focus on the most substantial activation area of the receptive matrix. Fig. 2.5a shows the highest 10% of values of this matrix, marked on the related raw depth images in five specific simulated training samples. The receptive fields of the 7500-iteration DRL model are compared with those extracted from the SL model [43]. As mentioned above, these two models consist of the same convolutional structures. We choose five specific samples located in the fallible area based on the trajectory heat map of the SL model, as shown in Fig. 2.4a. Before being transported to the *Softmax* layer, feature representations of the SL model were firstly transformed to five values related to the five commands in [43]. These values are listed with the action-evaluations estimated by the 7500-iteration DRL model in Table 2.4. For both of the models, the highest value corresponds to the optimal moving command.

Note that the area beyond the detection range of the *Kinect* camera is labelled as zero in the raw depth images. From Fig. 2.5a and Table 2.4, for the SL model, the moving



(a) Simulation environmental samples

(b) Real-world samples

Figure 2.5: The receptive fields of the feature representations extracted by CNN in both simulated environment samples and real-world samples. The purple area marked on the raw depth image represents the highest 10% of activation values. Both the supervised learning model (SL) and the 7500-iteration deep reinforcement learning model (DRL) are compared. The arrow at the bottom of each receptive image is the chosen moving command based on evaluations listed in Table. 2.4. The left column shows the RGB images taken from the same scenes, for reference.

command towards the deepest area in the range receives the highest output value naturally. This motivates the convolutional model to activate the further area of the scenes, especially the junction with the untracked white fields. In *S1*, the furthest reflection can help the robot avoid close obstacles. However, when there are multiple-level obstacles like in *S2* and *S3*, simply choosing the furthest part as the moving direction leads to collisions with nearby obstacles. Except for the furthest part, the DRL model also perceives the width of the route, both in the nearby area and the furthest area, as several horizontal cognitive stripes in the figure. That means the end-to-end deep reinforcement learning dramatically tunes the initial CNN weights from the SL model. From evaluations listed in Table 2.4, the DRL model not only helps the robot avoid instant obstacles, but also improves the traversable detection ability, like in *S4* and *S5*. When the route in *S4* is not wide enough to pass through, the DRL model chooses the fully turning moving command. However, the SL model always chooses the furthest area.

To prove the robustness of the trained model, five samples collected from a real-world environment by a *Kinect* camera mounted on a real *Turtlebot* are also tested, as shown in Fig. 2.5b. The related command evaluations for the SL model and the DRL model, as mentioned above, are listed in Table 2.4 as well. Note that, these real-world scenes are not included in the training datasets for the SL model [43] either. The receptive fields of the SL model are still mainly focused on the furthest area. Output values in Table 2.4 present the limited obstacle avoidance ability of the SL model for theses untrained samples. For the DRL model, even though only trained in the simulated environment, it keeps showing its ability to track the width of the route for real-world samples. In *R1* and *R2*, the trained DRL model successfully detects the traversable direction. In *R3*, it avoids the narrow space which is not wide enough to pass through. In *R4*, it chooses the optimal moving direction to turn fully left. However, in *R5*, when faced with irregular nearby obstacles which are not implemented in the training environment, it keeps tracking the width of the furthest area and fails to avoid irregular obstacles.

Another fact about the real-world tests is that the estimation of the action-value can reflect the future expectation to some extent. The estimation values of *R3* and *R4* listed in Table. 2.4 for all moving commands are obviously less than the values of other scenes. This corresponds to the higher probability of collision when there are nearby obstacles.

2.5 Conclusion

In this chapter, the utility of the deep reinforcement learning framework for robot obstacle avoidance is proved under end-to-end training. The framework comprises two parts, convolutional neural networks for feature representations and fully-connected networks for decision making. We initialized the weights of the convolutional networks by a previously trained model based on our previous work [43]. The deep reinforcement learning model extends the cognitive ability of mobile robots for more complicated indoor environments in an efficient online-learning process continuously. Analysis of receptive fields indicates the crucial promotion of end-to-end deep reinforcement learning: the feature representations extracted by convolutional networks are motivated substantially for the traversability of mobile robots both in simulated and real environments.

CHAPTER 3

CONCLUSION

In this thesis, we have considered sensorimotor learning for ground robot navigation. We targeted to find an efficient pipeline to learn and deploy cognitive navigation policies in unstructured, unfamiliar, and dynamic environments.

3.1 Discussion and Outlook

During this academic research, we have found more questions than answers in various fields. Currently, we have successfully applied model-free reinforcement learning and imitation learning in mobile robot navigation for both indoor and outdoor environments, considered the prediction and memory ability of neural networks and tackled the domain adaptation problem of the trained policy. However, there are still several less-addressed issues: (1) The training of model-free reinforcement requires a large number of samples. Thousands of pieces of computation hardware and highly parallelised and distributed threads become necessary, especially for deep reinforcement learning [2, 13]. (2) Without implicitly or explicitly predicting the future states of the dynamic environment [1, 5], it is still challenging to generate an efficient navigation strategy among crowded pedestrians. And (3) Effective metrics to measure the visual-based autonomous driving in the real world is still missing.

Here, we briefly summarise the related future works for both deep reinforcement learning and imitation learning for navigation policies considering the two aforementioned issues.

For deep reinforcement learning, model-based algorithms are generally considered to be sample efficient [9]. Predicting the forward image flow directly based on historical image states and actions was considered in [14]. They collected data through random motions of manipulators to estimate the visual-based dynamics model. This forward model was combined with model predictive control to finish several manipulation tasks. It can also be generalized to unknown objects. Further, background occlusions [12] and registrations with start and target images [11] were considered to solve longer-moving and more complicated tasks. Beyond leveraging the model with MPC, other research [37] has let the model-free methods first imitate the behaviour of a model-based method. Combining model-free and model-based methods, TRPO [41] was also improved with a learned dynamic model in [25]. It yielded a much more stable learning process compared with the vanilla model-based reinforcement learning methods. However, model-based reinforcement learning always lags behind the best model-free algorithms, especially when learning the model through high-dimensional approximators like deep neural networks. Employing uncertainty-aware dynamics models with sampling-based uncertainty propagation [8] has achieved comparable results to the best model-free algorithms on several challenging benchmark tasks. Extensions of model-based deep reinforcement learning on ground robot navigation would be exciting.

To predict the future state of nearby dynamic agents, learning a sensory-dynamics-model for autonomous driving is necessary. Considering the high dimension of RGB images, building this model on a top-view 3D lidar and first-person view semantic segmentation information through imitation learning may be a potential direction [39, 40]. Also, how to train a forward dynamics model efficiently with limited data is rarely considered. Xu *et al.* [32] introduced a novel algorithmic framework for model-based reinforcement learning algorithms with theoretical guarantees. This is particularly critical for mobile robot tasks. In terms of behaviour cloning, leveraging the implicit information but not just the raw sensor input may improve the efficiency of the model training. For example, gaze information has been proved to speed up the training of novice human learners. Imposing this kind of information [7, 28] may also encourage the evolution of imitation learning for navigation.

In Part **??**, we have to measure the *real-to-sim* pipeline for outdoor autonomous driving in another extreme simulated environment finally, because it is not trivial to achieve quantative results for this outdoor driving condition in the real world. An effective metric for the visual-based outdoor autonomous driving, especially for learning-based method, is still an open problem.

In general, research on sensorimotor learning is rapidly developing. In this thesis, we explored the sensorimotor learning from both theoretical and practice perspectives and deployed various learning-based navigation policy from different sensors for the first time, in both indoor and outdoor environments and both real and simulated worlds. To realize the target of *living with robots* and allowing the robot to achieve *human-like behavior*, sensorimotor learning will be expected to tackle various scenarios that we have not met before.

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. In *International Conference on Learning Representations*, 2018.

[3] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Active mobile robot localization. In *IJCAI*, pages 1346–1352, 1997.

[4] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active neural localization. In *International Conference on Learning Representations*, 2018.

[5] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. *arXiv preprint arXiv:1809.08835*, 2018.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018.

[7] Yuying Chen, Congcong Liu, Lei Tai, Ming Liu, and Bertram E Shi. Gaze training by modulated dropout improves imitation learning. *arXiv preprint arXiv:1904.08377*, 2019.

[8] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4754–4765. Curran Associates, Inc., 2018.

[9] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[10] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1329–1338, 2016.

[11] Frederik Ebert, Sudeep Dasari, Alex X. Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 983–993. PMLR, 29–31 Oct 2018.

[12] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 344–356. PMLR, 13–15 Nov 2017.

[13] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[14] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[15] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, July 2016.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, volume 1. MIT Press, 2016.

[17] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2829–2838, 2016.

[18] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[20] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[21] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[22] Dongshin Kim, Jie Sun, Sang Min Oh, James M Rehg, and Aaron F Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *2006 IEEE international conference on robotics and automation (ICRA)*, pages 518–525. IEEE, 2006.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[24] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1):47–63, 1991.

[25] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[26] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[27] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[28] Congcong Liu, Yuying Chen, Lei Tai, Haoyang Ye, Ming Liu, and Bertram E Shi. A gaze model improves autonomous driving. In *Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications*. ACM, 2019. to appear.

[29] Ming Liu, Francis Colas, Luc Oth, and Roland Siegwart. Incremental topological segmentation for semi-structured environments using discretized gvg. *Autonomous Robots*, 38(2):143–160, 2015.

[30] Ming Liu, Francis Colas, François Pomerleau, and Roland Siegwart. A markov semi-supervised clustering approach and its application in topological map extraction. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4743–4748. IEEE, 2012.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[32] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2019.

[33] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1928–1937, 2016.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[36] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L. Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 739–746. MIT Press, 2006.

[37] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[38] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[39] Nicholas Rhinehart, Kris M. Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[40] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[42] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[43] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *2016 IEEE/RSJ International Conference on Intelligent Robots*

and Systems (IROS), pages 2759–2764, Oct 2016.

[44] Lei Tai, Shaohua Li, and Ming Liu. Autonomous exploration of mobile robots through deep neural networks. *International Journal of Advanced Robotic Systems*, 14(4):1729881417703571, 2017.

[45] Lei Tai and Ming Liu. Mobile robots exploration through cnn-based reinforcement learning. *Robotics and biomimetics*, 3(1):24, 2016.

[46] Lei Tai and Ming Liu. A robot exploration strategy based on q-learning network. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 57–62, June 2016.

[47] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

[48] Ye Tao, Rudolph Triebel, and Daniel Cremers. Semi-supervised online learning for efficient classification of objects in 3d data streams. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2904–2910. IEEE, 2015.

[49] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR.

[50] Chris Xie, Sachin Patil, Teodor Moldovan, Sergey Levine, and Pieter Abbeel. Model-based reinforcement learning with parametrized physical models and optimism-driven exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 504–511. IEEE, 2016.

[51] Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[52] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *The European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[53] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[54] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378, Sept 2017.

[55] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.

APPENDIX A

LIST OF PUBLICATIONS

Journal Publications

1. Jingwei Zhang*, **Lei Tai***, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, Wolfram Burgard, "VR-Goggles for Robots: Real-to-sim Domain Adaptation for Visual Control", (* indicates equal contribution), *IEEE Robotics and Automation Letters (RA-L)*, 2019.

2. Peng Yun, **Lei Tai**, Yuan Wang, Ming Liu, "Focal Loss in 3D Object Detection", *IEEE Robotics and Automation Letters (RA-L)*, 2019.

3. **Lei Tai**, Shaohua Li, Ming Liu, "Autonomous Exploration of Mobile Robots through Deep Neural Networks", *International Journal of Advanced Robotic Systems (IJARS)*, 2017.

4. **Lei Tai**, Ming Liu, "Mobile Robots Exploration through CNN-based Reinforcement Learning", *Robotics and Biomimetics*, 2016.

Conference Publications

1. **Lei Tai**, Peng Yun, Yuying Chen, Congcong Liu, Haoyang Ye, Ming Liu "Visual-Based Autonomous Driving Deployment from a Stochastic and Uncertainty-Aware Perspective", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 4-8, 2019.

2. Yuying Chen*, Congcong Liu*, **Lei Tai**, Ming Liu, Bertram Shi "Gaze Training by Modulated Dropout Improves Imitation Learning", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 4-8, 2019.

3. Congcong Liu*, Yuying Chen*, **Lei Tai**, Haoyang Ye, Ming Liu, Bertram Shi, "A Gaze Model Improves Autonomous Driving", *ACM Symposium on Eye Tracking Research & Applications (ETRA)*, Denver, USA, June 25-28, 2019.

4. **Lei Tai**, Jingwei Zhang, Ming Liu, Wolfram Burgard, "Socially-compliant Navigation through Raw Depth Inputs with Generative Adversarial Imitation Learning", *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 21-25, 2018.

5. **Lei Tai**, Giuseppe Paolo, and Ming Liu, "Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancuver, Canada, September 24-28, 2017.

6. **Lei Tai**, Haoyang Ye, Qiong Ye, Ming Liu, "PCA-aided Fully Convolutional Networks for Semantic Segmentation of Multi-channel fMRI", *International Conference on Advanced Robotics (ICAR)*, Hong Kong, China, July 10-12, 2017.

7. **Lei Tai**, Shaohua Li, and Ming Liu, "A Deep-Network Solution Towards Modelless Obstacle Avoidence", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, October 9-14, 2016.

8. **Lei Tai**, Ming Liu, "A Robot Exploration Strategy Based on Q-learning Network", *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Angkor Wat, Cambodia, June 6-10, 2016.

Workshop Publications

1. Congcong Liu*, Yuying Chen*, **Lei Tai**, Ming Liu, Bertram Shi, "Utilizing Eye Gaze to Enhance the Generalization of Imitation Network to Unseen Environments", *International Conference on Machine Learning (ICML) Workshop*, June 14, Long Beach, USA, 2019.

2. Oleksii Zhelo, Jingwei Zhang, **Lei Tai**, Ming Liu, Wolfram Burgard, "Curiosity-driven Exploration for Mapless Navigation with Deep Reinforcement Learning", *International Conference on Robotics and Automation (ICRA) Workshop*, May 21, Brisbane, Australia, 2018.

Technical Reports

1. **Lei Tai***, Jingwei Zhang*, Ming Liu, Joschka Boedecker, Wolfram Burgard, "A Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation", (* indicates equal contribution).

2. Jingwei Zhang, **Lei Tai**, Joschka Boedecker, Wolfram Burgard, Ming Liu, "Neural SLAM: Learning to Explore with External Memory".

3. **Lei Tai**, Ming Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots".