Deep Learning Based Methods for Outdoor Robot Localization and Navigation

Doctoral Dissertation Reviewed by

Hosei University

Deep Learning Based Methods for Outdoor Robot Localization and Navigation

SIVAPONG NILWONG

ABSTRACT

The number of elderly people is increasing around the globe. In order to support the growing of ageing society, mobile robot is one of viable choices for assisting the elders in their daily activities. These activities happen in any places, either indoor or outdoor. Although outdoor activities benefit the elders in many ways, outdoor environments contain difficulties from their unpredictable natures. Mobile robots for supporting humans in outdoor environments must automatically traverse through various difficulties in the environments using suitable navigation systems.

Core components of mobile robots always include the navigation segments. Navigation system helps guiding the robot to its destination where it can perform its designated tasks. There are various tools to be chosen for navigation systems. Outdoor environments are mostly open for conventional navigation tools such as Global Positioning System (GPS) devices. In this thesis three systems for localization and navigation of mobile robots based on visual data and deep learning algorithms are proposed. The first localization system is based on landmark detection. The Faster Regional-Convolutional Neural Network (Faster R-CNN) detects landmarks and signs in the captured image. A Feed-Forward Neural Network (FFNN) is trained to determine robot location coordinates and compass orientation from detected landmarks. The dataset consists of images, geolocation data and labeled bounding boxes to train and test two proposed localization methods. Results are illustrated with absolute errors from the comparisons between localization results and reference geolocation data in the dataset. The second system is the navigation system based on visual data and a deep reinforcement learning algorithm called Deep Q Network (DQN). The employed DQN automatically guides the mobile robot with visual data in the form of images, which received from the only Universal Serial Bus (USB) camera that attached to the robot. DQN consists of a deep neural network called convolutional neural network (CNN), and a reinforcement learning algorithm named Q-Learning. It can make decisions with visual data as input, using experiences from consequences of trial-and-error attempts. Our DQN agents are trained in the simulation environments provided by a platform based on a First-Person Shooter (FPS) game named ViZDoom. Simulation is implemented for training to avoid any possible damage on the real robot during trial-and-error process. Perspective from the simulation is the same as if a camera is attached to the front of the mobile robot. There are many differences between the simulation and the real world. We applied a marker-based Augmented Reality (AR) algorithm to reduce differences between the simulation and the world by altering visual data from the camera with resources from the simulation.

The second system is assigned the task of simple navigation to the robot, in which the starting location is fixed but the goal location is random in the designated zone. The robot must be able to detect and track the goal object using a USB camera as its only sensor. Once started, the robot must move from its starting location to the designated goal object. Our DQN navigation method is tested in the simulation and on the real robot. Performances of our DQN are measured quantitatively via average total scores and the number of success navigation attempts. The results show that our DQN can effectively guide a mobile robot to the goal object of the simple navigation tasks, for both the simulation and the real world.

The third system employs a Transfer Learning (TL) strategy to reduce training time and resources required for the training of newly added tasks of DQN agents. The new task is the task of reaching the goal while also avoiding obstacles at the same time. Additionally, the starting and the goal locations are all random within the specified areas. The employed transfer learning strategy uses the whole network of the DQN agent trained for the first simple navigation task as the base for training the DQN agent for the second task. The training in our TL strategy decrease the exploration factor, which cause the agent to rely on the existing knowledge from the base network more than randomly selecting actions during the training. It results in the decreased training time, in which optimal solutions can be found faster than training from scratch.

We evaluate performances of our TL strategy by comparing the DQN agents trained with our TL at different exploration factor values and the DQN agent trained from scratch. Additionally, agents trained from our TL are trained with the decreased number of episodes to extensively display performances of our TL agents. All DQN agents for the second navigation task are tested in the simulation to avoid any possible and uncontrollable damages from the obstacles. Performances are measured through success attempts and average total scores, same as in the first navigation task. Results show that DQN agents trained via the TL strategy can greatly outperform the agent trained from scratch, despite the lower number of training episodes.

ACKNOWLEDGEMENT

This thesis serves as a trophy for challenges in its development. It takes not only a strong determination, but also a cooperation among friends and family, and a courage of thinking beyond boundaries of sanity. Development of this thesis proved that despite the game addiction, those disadvantages can be changed into advantages instead.

First and foremost, I would like to express my sincere gratitude to my advisor, **Professor Dr. Genci Capi**, for his guidance, inspiration, endless support, and kind hospitality. He has supported me since the days before I moved to Japan. He helped me from finding a place for me to live in, creating appropriate atmosphere for the research, and kindly response to all my requests related to the research. He was the only person who did not go against any of my bizarre ideas during the research, and yet support those ideas to his fullest potential. His guidance was the most essential part to complete all works during my Ph.D. research. This includes everything from readings, writings, making reviews, presentation styles, and sharpening ideas into the field full of competitions. I cannot find any words suitable enough to thank him for all his efforts.

I would like to thank all members of Assistive Robotics Laboratory, Hosei University. It was a great experience knowing and working along with each lab member. As I began to understand styles of lab members, my life in laboratory become easier with all support I could ask for. My thanks go to my fellow lab members who I am not thinking as friends, but more like a big family. Especially, my sincere gratitude to **Dr. Delowar Hossain**, who supported me in both daily life and the research.

I would like to express my deepest gratitude towards my incomplete family of 4 people, including my mother, my aunt, my uncle, and my grandmother. Despite being incomplete, they always supported me both physically and mentally. They encouraged me to work to my fullest capabilities, as they always believed that I would never give up or give in.

This thesis would not be in its current state if not for inspirations from three video games, including **NieR:Automata** by **PlatinumGames**, **DOOM (2016)** by **Bethesda Softworks**, and **WARFRAME** by **Digital Extremes**. Inspirations from these games lead to the new dawn in my robotics research.

Hosei University **Sivapong Nilwong**

April 2020

DEDICATION

“For the glory of mankind”

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Mobile robots are robots with locomotion mechanisms that grant capabilities of transporting themselves from places to places. Importance of mobile robots rises as demands for service robots increase across the globe, especially autonomous mobile robots. The autonomous mobile robots usually packed with sensors and controllers which help them to understand their operating environments. They can navigate automatically using maps, dynamic algorithms, or any available data, instead of fixed routes and algorithms. Applications of mobile robots spread among various field. Mobile robots can be seen operating in daily life such as household caring, vacuum cleaning, and elderly supports. Industrial sections also employ mobile robots in their business. Robots operating in industrial settings increase manufacturing efficiency, while providing simplicity to their operators. Examples of industrial applications which implemented mobile robots include medicine delivery in hospitals, warehouse managements, scientific experiments, and maintenance of machines and structures.

Domains of mobile robots can be separated by their operating environments into two domains: indoor robots and outdoor robots. Indoor robots are robots which operating in indoor areas such as areas inside houses, rooms, and buildings. Outdoor robots operate in outdoor areas such as fields, farms, and streets. Regardless of the domains, success of any mobile robots requires one significant principle: the navigation. Mobile robots require proper navigation systems to propel themselves to their goals and perform according to their tasks. Goals of mobile robots can be either objects or places. In order to assist humans to their full extents, suitable navigation tools need the be chosen for robot navigation systems. Each environment poses various challenges for navigation systems, and choices of navigation tools are varied. There are conventional navigation tools implemented for navigation in daily life basis. Examples of these conventional navigation tools include the global positioning systems (GPS) and magnetic compass. Mobile robots also applied these conventional tools for their navigation systems, especially in outdoor environments. It is known throughout all fields of study that conventional navigation tools have a considerable amount of limitations. Many device manufacturers provide combinations of different tools and sensors to overcome these limitations. For instance, smartphones are provided with phone signals to improve accuracy of the embedded GPS.

Mobile robots can complete their tasks effectively with accurate navigation systems. Navigation systems also require the ease of maintenance. While increasing number of sensors and tools can reduce limitations and increase accuracy, it also increases the complexity with addition of conditions. Reducing complexity of the navigation systems can result in decreasing chances of programming errors that can cause severe problems to robots. Mobile robots with simple and accurate navigation systems are mostly preferred for practical applications, particularly human-supporting applications.

1.2 Motivation

Many countries around the globe are moving into the ageing society. This circumstance lead to an increasing demand of service mobile robots. As outdoor activities can benefit elderly people in many ways, creating mobile robots for supporting elderly in outdoor environments can promote the sustainable life. For instance, a robot can accompany the elderly to places and guide them back to their home, as illustrated in Figure 1.1. Apart from specified tool designed for their designated tasks, mobile robots essentially require navigation systems to propel themselves to their goals and finish the tasks. Most of mobile robots in outdoor environments are applied with conventional navigation tools such as GPS and magnetic compass. These conventional tools are simple and efficient at some levels, which make them suitable for most of navigation tasks. In some applications of mobile robots, conventional navigation tools are not precise enough

Figure 1.1: A mobile robot accompany the elderly across the street.

for the tasks. This led to additions of sensors to reduce errors. However, additions of sensors also increase complexity to the navigation systems, with more conditional expressions from the sensors.

The addressed problems inspired the motivation of this work. Normally, the use of multiple navigation tools causes larger codes in the programs. Natures of outdoor environments are complex and unpredictable, which also add more complexity and difficulties to the navigation systems. Outdoor robot navigation systems suffer from latency, inefficiency, and more chances of programming errors from employments of conventional navigation tools. Therefore, reducing the use of conventional tools and algorithms is crucial for advancements in outdoor robot navigation. This thesis focuses on the development of an automated outdoor localization and navigation system for mobile robots with less implementations of navigation tools. The developed localization system is expected to allow the mobile robot to know its location in outdoor environments, while the developed navigation system is expected to guide the robot to its goal automatically with the least number of navigation tools possible.

1.3 Objectives

The main objective of this work is to develop outdoor localization and navigation systems based on deep learning approaches. The localization system is planned to be developed using a supervised deep learning algorithm, while the navigation system is based on an unsupervised algorithm. Goal of both systems are to reduce large number of sensors required for the localization and navigation tasks to minimum, which not only reduce the number of conditional statements in the robot programs caused by each sensor, but also lower the risks from sensors that are occasionally unreliable. Camera is the key of both systems. The localization system should be able to know robot locations in the real world using only an image, replacing GPS devices and magnetic compasses. The navigation system should be able to use visual data to guide the robot to its goal within short distances, where errors of typical navigation devices cause them to be unavailable.

The development of the deep learning-based outdoor localization system has the goal of reducing, or even replacing traditional sensors required for knowing current locations of the robots. As the robot learns from metadata from images of landmarks, it gradually knows locations of images from these data. This process resembles the way humans know their locations from landmarks. Therefore, the use of sensors can be reduced to the point where only a camera is required for localization. Thus, reducing programming burdens caused by additional sensors.

The objective of the unsupervised outdoor navigation system is to implement an unsupervised deep reinforcement learning algorithm for outdoor navigation tasks. Deep reinforcement learning algorithms usually do not require supervisions from human, as these algorithms can learn their tasks through trial-and-error process. It can reduce number of data that humans have to feed for training the robot and reduce number of conditions during the navigation. With on visual data as input, the navigation system can guide the robot with only a camera, in which risks from unreliable sensors are lower.

The last objective of this thesis is to employ a transfer learning strategy for multiple navigation tasks of the mobile robots. We train the first deep reinforcement learning model for the simple task of reaching the goal. The trained model is transferred to train a model for the second task that is more difficult with obstacles. We reduce exploration factor during the transfer learning. The reduced exploration factor causes the second model to utilize prior knowledge in the first model during the learning process. It can help the second model find the optimal navigation solutions for the task faster, with better navigation performances.

1.4 Contributions

The progress of this thesis is separated into several phases. The first phase includes the development of a deep learning-based localization system, which trained with data from the real world. The second phase involves the development of a deep reinforcement learning agent for navigation, using an algorithm called Deep Q Network (DQN). The development of the DQN agent in the second phase focuses on the simple task of reaching the goal. The second phase includes the creation of simulation environments, the learning process of DQN agents through trial-and-error attempts, and experiments in the simulation and the real environment. The final phase of this work focuses on the implementation of a transfer learning strategy, which applied the agent trained in the second phase for more advanced tasks. The address issues lead to main contributions of this thesis, which are detailed in following subsections.

1.4.1 Creating landmark-based outdoor robot localization system using deep learning algorithms

Deep learning algorithms have been widely implemented for the tasks of object detection and recognition. Convolutional Neural Network (CNN) based algorithms, that are specialized with two-dimensional data, are especially implemented for the detection tasks. These algorithms are accurate with proper training data. From performances of CNN based algorithms, they can be applied for further tasks, including localization. Humans can know places just by watching signs or landmarks nearby. As deep learning methods are modeled after human brains, deep learning models can mimic one of those abilities, the ability of knowing places from landmarks. Models can be trained with proper geolocation data and landmarks that can be detected in images. This results in following advantages:

a) The sensors for robot localization can be reduced to only a single camera. It can reduce risks from unreliable sensors which have performances depend on environmental conditions.

b) As the number of sensors reduced, programming burdens from conditional statements caused by each sensor are also reduced.

c) The localization system contains the object detection part that can be reuse extensively for countless purposes.

1.4.2 Implementing Deep Q Network (DQN) agents for simple outdoor navigation tasks

Deep Q Network is a reinforcement learning algorithm which make decisions from visual data. DQN is proved to be formidable for making decisions in various implementations. As a reinforcement learning algorithm, it trains its agents through trial-and-error process. DQN agents are almost always trained in simulations to prevent possible damage to real machines. Agents must work in real environments with knowledge learned from simulation environments. We employ a simple marker-based Augmented Reality (AR) algorithm to augment robot vision to be similar to what the robot agent learned in simulations. Contributions from this section include:

a) It makes the robot make decisions from visual data on its own, reducing sensor dependence and computation complexity of navigation algorithms.

b) There is no need to input training data to the system, since the DQN is unsupervised learning. This saves time and efforts in preparing training datasets.

c) The implemented AR algorithm is based on object detection. Choices for object detection algorithms are countless. It can be either simple or complex, depending on choices of users.

1.4.3 Transferring knowledge from DQN agents for the training of agents for more advanced tasks

In order to navigate the robot in specified tasks, navigation agents can be simply created for each task. However, many navigation tasks share close similarities to each other. Knowledges from agents which were created for simple tasks can be added to new agents that have to perform advanced tasks. Therefore, we apply a transfer learning strategy for the training of advanced DQN agent using knowledge from the agent for the simple navigation task with the same objective. There are advantages from transferring knowledge over creating new agent from scratch, as follows:

a) It replaces the random DQN parameters with some parameters form the old agent. Some explorations in trial-and-error process are more optimized.

b) It improves training performances of DQN agents. New agents can utilize existing knowledge during training, finding optimal solutions more effectively.

c) It reduces a significant amount of training time by reducing some parts of trial-and-error exploration process.

1.5 Thesis Outline

The structure of our thesis reflects the methodological steps and contributions of our work. This thesis is divided into eight chapters. Description of each section is provided to simplify the perception of this thesis.

- ***Chapter 1*** introduces this thesis. Background, motivation, objectives, and contributions of the thesis are enclosed in this chapter.

- ***Chapter 2*** presents a literature review of related works. Significant and distinguished works including vision-based localization and navigation systems, deep reinforcement learning algorithms, and transfer learning strategies are explained. This chapter also includes previous works with implementations of simulation platforms on robots.

- ***Chapter 3*** illustrates an overview of the proposed systems for outdoor mobile robots and their flows. Localization tasks and navigation tasks are described. The applied transfer learning strategy for the employed DQN is also explained

- ***Chapter 4*** displays results from our experiments. There are three sections in this chapter, including the results from the deep learning based outdoor localization system, results from the DQN navigation system for outdoor mobile robots, and results from an improvement of DQN with the transfer learning strategy.

- ***Chapter 5*** concludes the thesis and discuss future works which can be evolved or applied from this research.

CHAPTER 2

RELATED WORKS

Various elements are involved in developing systems for localization and navigation for mobile robots in outdoor environments using visual data and deep learning. Such elements include robot vision systems, navigation algorithms, deep learning algorithms. For some deep learning algorithms, simulation environments are also included in their development processes. Literature review in this chapter consists of four main sections, including existing vision-based systems for mobile robot localization and navigation, deep learning approaches for vision systems, deep reinforcement learning algorithms, and existing transfer learning strategies for deep learning.

2.1 Vision-Based Localization and Navigation Systems

Navigation is one of core principles that mobile robots need in order to accomplish their given tasks. Success in navigation requires many factors. One of significant factors is the localization, which allows the robots to be able to determine their positions in the environments [1]. Typically, navigation systems of mobile robots can rely on conventional navigation tools, same as humans. Examples of these tools include Global Positioning Systems (GPS) devices and magnetic compasses, which are widely used for localization in navigation systems. Due to their simplicity, these conventional tools are used extensively in different mobile robot tasks, some of which include the patrolling robot in outdoor environments in [2] that employed GPS and robot odometer in its navigation system, the integrated navigation system from [3] which combines differential magnetic compass and GPS with an algorithm based on federated Kalman filter, and the simple mobile robot navigation system in [4] that introduced with low-cost GPS receivers. Despite the large number of implementations of conventional navigation tools, these tools have been proved to possess reliability problems. One significant instance is the GPS. According to the report [5] cited by the US government, typical GPS devices applied in smartphones have an average error of 4.9 m. The report [6] added that GPS devices have more errors, or even unavailable in environmental conditions that can block GPS signals.

Different approaches were proposed to improve or replace the use of these conventional tools. Examples of these approaches include the General Packet Radio Service (GPRS) signals [7], implementations of GPRS signals and remote sensing networks [8–9], the robot wheel odometers [10], and the wireless local area networks (WLAN) for robot navigation [11]. Among the available navigation and localization approaches, vision-based systems are widely implemented for the mobile robots. The vision-based systems introduce the use of vision systems such as cameras to localize and navigate the robots in their environments. In the simplest approaches, vision systems are employed together with image processing algorithms, in which the approaches can start from edges and lines detection [12–15]. As more image processing and computer vision algorithms were proposed, so did the vision-based localization and navigation algorithms. Localization of mobile robots and their navigation could be accomplished through features of different properties in visual data [16–25], or bonded together with feature-based computer vision algorithms such as the Scale-Invariant Feature Transform (SIFT) [26–31] and the Speeded Up Robust Features (SURF) [32–36].

In recent years, vision-based systems were more oriented to deep learning-based algorithms, due to their capabilities of learning and understanding abstract relationships in images. These capabilities are lacked in image processing algorithms and feature detection algorithms. Some examples of such deep learning for visual navigation include the use of Convolutional Neural Networks (CNN) in [37–39]. Vision-based deep learning approaches and their applications into the robot localization and navigation tasks are described in the next section: Deep Learning Approaches for Vision Systems.

2.2 Deep Learning Approaches for Vision Systems

Deep learning is part of a family of machine learning methods, which based on artificial neural networks and representation learning. It allows computational models with multiple processing layers to learn representation of data with multiple level of abstraction [40]. Representation learning in deep learning allows a machine to be fed with raw data and to discover representations of the fed data, according to their desired goals. There are various domains that deep learning can be employed to improve, some of which include visual object recognition, object detection, speech recognition, and natural language processing [41].

There are many deep learning approaches that proposed specifically for the tasks with visual information. Examples of such approaches include Convolutional Neural Networks (CNN) and Deep Belief Networks (DBNs). Though many works suggested that the simple Feed-Forward Neural Network (FFNN) can handle tasks with visual information such as [42], it required tough efforts on the preprocessing of the data when used.

DBN is a probabilistic generative model that introduced with a novel method of pre-training neural networks, using the Restricted Boltzmann Machine (RBN) [43]. RBN was employed to initialize weights in deep autoencoder networks. DBN performed well with impressively low amount of errors in MNIST handwritten digit recognition tests. However, DBN had scalability problems, which make it inferior to 2D structure of images. Later variations of DBN solved this problem by applying the probabilistic max-pooling technique, became the Convolutional Deep Belief Networks (CDBN) [44].

CNN is by far an approach among the most implemented deep learning instances for computer vision tasks. Its inspiration can be traced back to the year 1962, as Hubel and Wiesel [45] proposed receptive fields in visual cortex of the cat. Inspiration from the animal visual cortex creates different vision-based neural network models. The Neocognitron by Fukushima in 1980 [46] is directly inspired by this visual cortex, with the neural network model that include “S-cells” which represent simple receptive fields, and “C-cells” that represent complex receptive fields of the animals. Lecun et al. proposed the CNN in 1998 [47]. It combines three ideas of network architecture, which are the local receptive fields, shared weights, and spatial subsampling. CNN consists of convolutional layers which can extract features from the input data with 2D shapes. It performed greatly for handwriting recognition tasks. CNN was evolved into many different deep learning approaches regarding their implementations. For instance, object detection task evolved

CNN into regional-based CNN algorithms. Girshick et al. proposed the method called R-CNN in 2014 [48]. It uses the selective search algorithm to extract regions from the image. These regions are called “region of proposals”. The extracted regions are processed in CNN to generate CNN features. Theses features are fed to a support vector machine (SVM) to classify the object in the region. R-CNN is practically slow. Thus, an upgrade for R-CNN called Fast R-CNN was proposed [49]. Instead of generating regions first, CNN is first used to generate a convolutional feature map. Then the selective search algorithm is applied to identify region of proposals. These regions are processed with a fully connected layer, identifying the object and regress values of bounding boxes of the objects at the same time. Fast R-CNN received another upgrade called Faster R-CNN [50]. It replaces the selective search with the Region Proposal Network (RPN). RPN is used to predict region proposals instead of searching. Thus, greatly reduced the detection time. Another CNN-based object detector is the algorithm named You Only Look Once (YOLO) [51]. YOLO uses a single CNN to predict both bounding boxes and class probabilities of the objects from full images. It performed very fast with an acceptable amount of errors. Though it has some problems with small objects in images and has less accuracy than the Faster R-CNN.

2.2.1 Vision-Based Deep Learning for Object Detection and Recognition

Vision-based deep learning approaches mostly focused in the tasks of object detection and recognition. Object detection and recognition applications started from the simplest model such as the FFNN. Khasnobish et al. [42] used FFNN to recognize shapes of objects from tactile images which are the consequence from human touching such objects. Jänen et al. [52] detected and tracked multiple objects in camera images with FFNN. Rázuri et al. [53] recognize human emotions within facial expression using an artificial neural network with the structure of FFNN. Despite being available available on the simple FFNN, all object detection and recognition algorithms proposed for their tasks require a significant amount of preprocessing during their operation time.

For more advanced deep learning approaches for vision tasks, DBN is greatly implemented for object detection and recognition. Chen et al. [54] applied DBN to detect aircrafts from remote sensing images. Multiple thresholds were used to extract segments that have the possibility to be aircrafts. The DBN then processed the segments to recognize aircrafts in those segments. Diao et al. [55] also detected objects from remote sensing images. In this case, however, DBN was trained directly with raw object images and saliency maps. Liang et al. [56] employed two DBNs to recognize and estimate poses of 3D objects detected by an object detector based on K-mean clustering. Zhao et al. [57] detected drowsiness of drivers using facial landmarks and facial textures as inputs for the DBN. Kamada and Ichimura [58] improved the DBN by optimizing RBM through the neuron generation-annihilation algorithm. The improved was tested on the Chest X-ray benchmark and received better classification and localization results than CNN methods.

Object detection and recognition tasks for deep learning generally include the implementations of CNN. It is widely implemented for detection and recognition tasks. We will cover some of the examples, as there are more than a thousand of CNN implementations available. Back in 2000s, the tasks of the CNN are mostly for classification purposes. Chen et al. [59] applied the CNN for the classification of faces and license plates. Input sizes of the CNNs are static, and the separate networks needed to be trained for each type of the classification. Szarvas, Sakai, and Ogata [60] combined a LIDAR and a CNN to detect pedestrian. LIDAR was employed to propose candidates for the regions of interest, while CNN detected pedestrian in those regions. In the following decade, CNN approaches were greatly implemented and improved. There were different CNN settings proposed, some changed the network name to fit in their tasks, while some employed the word CNN as the network name. Zong et al. [61] applied different CNN structures to detect human presences in millimeter wave images. Mane and Mangale [62] utilized CNN for tracking moving objects in images. Haque, Lim, and Kang [63] employed ResNet with residual learning and the CNN with the structure named VGG network in the task of object detection. The CNN evolved for object detection and recognition tasks into R-CNN [48], Fast R-CNN [49], and Faster R-CNN [50]. Applications for object detection using these CNN-based detectors were opened into more possibilities. Li et al. [64] employed the Faster R-CNN for detecting aircrafts in remote sensing images. The method did not require much preprocessing during its operation time. Saleh et al. [65] utilized Faster R-CNN for detecting pedestrian in synthesized depth images. Zhang et al. [66] detected pedestrians in images from security cameras using Faster R-CNN. Wang et al. [67] detected ships in images with foggy weather. A CNN-based network was also proposed to classify scenes in images.

2.2.2 Vision-Based Deep Learning for Robotics Tasks

Vision-based deep learning approaches are implemented in different sections of robotics works. These vision algorithms are always implemented with the use of cameras, as they can help robots understand representations of things that visualized to the robots. One of the most proposed tasks for robots with vision systems is object detection, since vision systems can help robots identify objects better than using combinations of different sensors. For instance, robots can detect objects using LIDAR [68]. However, shapes of visible objects are limited, and require a plenty of calculations. For the tasks of detection and recognition in robots, different deep learning applications were proposed. Zhihong et al. [69] employed Faster R-CNN with VGG-16 as its CNN for the task of garbage sorting. The robot manipulator acquired visual data of the conveyor belt filled with garbage. Faster R-CNN detected the garbage and sort them with the manipulator. Zunjani et al. [70] predicted optimal grasping locations of different objects based on their intents. The algorithm for detecting object and predict grasping locations is the CNN-based algorithm called Mark R-CNN. Yoshimoto and Tamukoh [71] employed an algorithm based on the VGG-16 model of CNN to recognize objects through the 3D Kinect camera for service robots. Akbar et al. [72] used CNN to detect runways for Unmanned Aerial Vehicles (UAV). Chen et al. [73] applied the deep learning model called ENet to detect obstacles in outdoor environments through robot cameras. The ENet contains convolutional layers, similar to CNNs. Ibrahim Khalilullah et al. [74] detected roads in images using the Deep Belief Neural Network (DBNN). This road detection helped the robot for its navigation in outdoor environments. Le, Huynh, and Pham [75] addressed the problem of human-robot interaction by applying Mark R-CNN to help localize human faces in images. Konoplich, Putin, and Filchenkov [76] combined CNN and multilayer perceptron (MLP) for the task of vehicle detection in UAV images. Budiharto et al. [77] detect objects in images from the camera attached to the quadcopter drone using the CNN-based Single Shot Detector (SSD). Chao, Chen, and Xiao [78] applied Faster R-CNN to detect objects and their grasping points for the robot manipulator with five fingers. Nuzzi et al. [79] also implemented Faster R-CNN for recognition. Their work used Faster R-CNN to detect hand gestures from RGB images taken from a Kinect camera.

Vision-based deep learning algorithms also involved in different designs of the robots, some of which include the use of the CNN-based algorithm to determine actions of the robot for folding clothes [80], and the design of the shop assistant robot using CNN in the Robot Operating System (ROS) platform [81]. As for the tasks of determining actions of the robot from visual data, many approaches moved on to the deep reinforcement learning.

2.3 Deep Reinforcement Learning Algorithms

In order to cover the basics of the deep reinforcement learning, we have to start from its predecessor: the reinforcement learning. Reinforcement learning is an area of machine learning that focuses on how artificial agents take actions in their environments to maximize the cumulative rewards [82]. Reinforcement learning algorithms can be modeled basically in the form of Markov Decision Processes (MDP), in which decision makers or agents find optimal solutions through processes that are partly random [83]. The learning process of reinforcement learning requires a balance between the exploration and exploitation. Exploration refers to the process where the agents explore consequences of actions by randomly selecting actions. Exploitation, however, is the process where agents "use" their known knowledge and perform actions which might be the optimal solutions. The method of ε-greedy is typically employed to balance the amounts between exploration and exploitation. The agents applying the ε-greedy method decide whether to explore or exploit based on the probability value stored in ε [84].

Reinforcement learning algorithms can be separated into two categories, according to their learning policies. The first category is the off-policy algorithms. The term "off-policy" means the algorithms learn with no regard to any policy. The only policy that off-policy reinforcement learning algorithms consider is the way to maximize cumulative reward [85]. One significant instance of the off-policy reinforcement learning algorithms is the Q-Learning [86]. Agents in Q-Learning use a table to store quality values (Q values) of actions in each state. The quality values are calculated through the Bellman equation. Actions which result in the maximum total rewards are performed in Q-Learning. The second category is the on-policy algorithms. On the contrary to off-policy algorithms, on-policy algorithms learn the to optimize values of the policies which are being carried out by the agents. An example of the on-policy algorithms is the SARSA [86]. While SARSA employs Q values same as the Q-Learning, it also includes the current policy of actions into the calculations of Q values.

Reinforcement learning algorithms mostly have scalability problems, they cannot work in large or complex environments. Deep reinforcement learning algorithms are proposed to solve these problems by fusing deep learning with reinforcement learning. A significant milestone in deep reinforcement learning algorithm is the introduction of the Deep Q Network (DQN) [87]. It combines the CNN with the Q-Learning. Therefore, it can take visual inputs that are considerably too large for Q-Learning. DQN was able to play different video games at human-level performances using only visual information from the video game screen. Another deep reinforcement learning algorithm is the Asynchronous Advantage Actor Critic (A3C) that combines deep learning and the actor-critic together with asynchronous running scheme [88]. DQN has its improvements as the Double DQN [89] and the Dueling DQN [90], which utilize multiple instances of DQN during the learning process.

2.3.1 Applications of Deep Reinforcement Learning for Robots

Performances of DQN for playing video games in [87] reveal capabilities of deep reinforcement learning algorithms in making decisions from visual data. Many robotics systems struggle with a large number of conditions that the robots have to deal with. Deep reinforcement learning algorithms are employed in various robotics works that require decent decision makers. Chen and Dai [91] proposed a control policy for the grasping tasks with the robot manipulator. The control policy was based on the DQN, while an additional CNN detector was attached to the control system to detect objects for grasping. Bejar and Morán controlled the autonomous truck-trailer systems in their backing movements using the Deep Deterministic Policy Gradient (DDPG) [92]. Özaln et al. [93] controlled the locomotion of the humanoid robot through the visual information from the camera. They employed the Double Dueling Deep Q Network (D3QN) and the DQN for controlling the robot. Kim et al. [94] applied DQN for the wheeled robots that play soccer.

Seo, Kim, and Kim [95] employed DQN to the humanoid robot for the task of push-recovery that help balancing the robot. Xue et al. [96] proposed a method for collision avoidance in mobile robots along with navigation capabilities, using the Double DQN. Bui and Chong [97] controlled speech volume in social robots using DQN. Lobos-Tsunekawa et al. [98] navigated biped humanoid robots through visual information. They controlled the robot using the DDPG algorithm and the Long-Short Term Memory (LSTM). Navigation is one of the tasks that widely employed deep reinforcement learning algorithms, as they are less restricted by robot sensors, places, or other environmental conditions. Deep reinforcement learning was implemented throughout almost every aspect for robot navigation, from path planning to locomotion [99–109].

2.4 Transfer Learning for Deep Learning

The ability of transferring knowledge across tasks is inherited in humans among generations. It can be implied that a man can learn how to ride a motorcycle better, if he knows how to ride a bicycle. The similar strategies are implemented in learning and deep learning algorithms. In learning algorithms, generalization and the insufficient data problems are needed to be solved [110]. Inductive transfer mechanism is proposed to overcome these problems [110–111]. It can improve generalization problems by leveraging specified information of related tasks. Different knowledges are transferred in the category of vision-based algorithms, with some of the most popular include the feature representation transfer and the classifier-based knowledge transfer [112].

Vision-based deep learning algorithms widely employed transfer learning into their implementations. The main reason is the insufficient training data. It is evident from some works that deep learning for computer vision tasks require a large amount of data in training, such as [113] that might require more than half a million datasets. Transfer learning helped vision-based deep learning with this data issue. Huang et al. [114] replaced layers of several the pre-trained CNN models, then re-trained the models for the task of plant classification. Kulkarni et al. [115] replaced the CNN inside the Faster R-CNN with the pre-trained Inception V2 model. Their Faster R-CNN model can detect traffic lights with better accuracy. Li et al. [116] classified protein patterns in human cells using the CNN-based model that acquired some parts from the pre-trained Inception V3 model. Pelletier et al. [117] applied the pre-trained ImageNet model to initialize convolutional layers in their classifier of marine resources. Dalal and Moh [118] transferred the deep learning model trained with the COCO dataset to be used on the large dataset that might require approximately five million iterations of training. Huber-fliflet et al. [119] used 13 convolutional layers of the pretrained VGG16 and replaced fully connected layers. The model was re-trained with the fine-tuning strategy for detection tasks of legal documents. Sferrazza and D'Andrea [120] applied transfer learning for vision-based tactile sensing. Their method transferred representations of features from one tactile sensor across different settings.

2.4.1 Transfer Learning for Deep Reinforcement Learning and Robots

While not only in the field of big data and computer vision, the implementations of transfer learning found themselves in reinforcement learning algorithms and their applied tasks, including the robotics tasks. In reinforcement learning, transfer learning can be used to transfer parameters of the same model structure [121]. Shao, Zhu, and Zhao [122] employed a transfer learning strategy to a reinforcement learning model for playing StarCraft in different scenarios. They trained a reinforcement learning model for the source task, then transfer the trained knowledge to other tasks. In robotics tasks, transfer learning can be applied to transfer knowledge from one task to another, in which the source tasks are usually simpler such as mentioned in [123].

The significant implementation of transfer learning for reinforcement learning algorithm in robotics works is the transferring of knowledge between the simulation and the real world. Typically, robots with reinforcement learning algorithms are preferred to be trained in simulations, since the learning process contains the exploration with random actions. These random actions can harm robot parts. Various methods were proposed for transferring the knowledge between the simulations and real environments, in which many of them include the transfer using the fine-tuning strategy [124–130].

CHAPTER 3

METHODOLOGY

The eruption of deep learning algorithms enables more possibilities for implementations of vision systems into robotics works. Many deep learning algorithms can support human in tasks related to visual data with human-level performances. As discussed in chapter two, there are various methods of deep learning available, even for visual data. Different advantages and disadvantages come with different deep learning instances at different circumstances.

In this chapter, three systems are presented regarding to their tasks. The first system is the localization system for outdoor mobile robots using the Faster Regional-Convolutional Neural Network (Faster R-CNN) and visual data in the form of images. Faster R-CNN can detect landmarks in images, in which landmarks can be utilized for determining the robot locations in real environments. The second system is the navigation system for mobile robots in outdoor environments using the deep reinforcement learning algorithm called Deep Q Network (DQN) with visual data from a single camera. DQN can be trained without input data, as it learns through trial-and-error experiences. The last system is an upgrade for the second system, where DQN agents can be trained using past experiences from other agents which do similar tasks.

3.1 Landmark-Based Outdoor Robot Localization with Faster Regional-Convolutional Neural Network

Faster Regional-Convolutional Neural Network (Faster R-CNN) is a deep learning algorithm specifically for visual object detection purposes. It is derived from its predecessor, the Fast Regional-Convolutional Neural Network (Fast R-CNN), which is also a deep learning algorithm for object detection based on the Convolutional Neural Network (CNN). Input for any Faster R-CNN models is in the form of an image. After the image is processed throughout the structure of the Faster R-CNN, outputs are generated. These outputs include bounding boxes of detected objects, corresponding labels of the objects, and detection scores for each detected object.

During the application for the task of landmark-based localization proposed in [131], the Faster R-CNN model can be trained to detect landmarks. Faster R-CNN model can generate bounding boxes of landmarks along with their corresponding labels. These properties of landmarks generated from the trained Faster R-CNN model can be used to determine locations of mobile robots. From the system flows illustrated in Figure 3.1, image of a location is input through the camera into the Faster R-CNN. The bounding boxes, labels, and detection scores of detected landmarks are generated. These generated properties are then processed to acquire locations and orientations of the robot in the form of latitude and longitude coordinates, along with the compass orientation. A simple Feed-Forward Neural Network (FFNN) is employed to process detect landmarks and their properties which generated from the Faster R-CNN. The FFNN is employed since it is

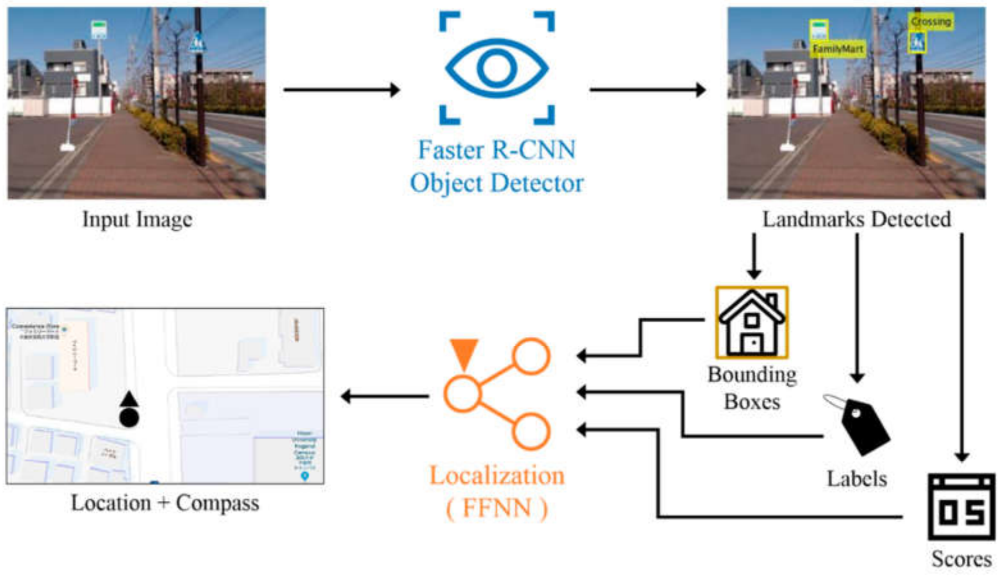

Figure 3.1: Flows of the proposed landmark-based localization system.

simple. It can determine locations and orientations from landmark properties, which are difficult for simple conditional programming statements, while keeping its simplicity through the learning process that has less human interference. Further details and reason for implementations of two core algorithms for the localization system, including Faster R-CNN and FFNN, are described in following subsections.

3.1.1 Faster R-CNN for Landmark Detection

As mentioned prior in this chapter, Faster R-CNN is a deep learning-based object detection algorithm that derived from Fast R-CNN and CNN. The Faster R-CNN contains two main parts, including the Fast R-CNN and the Region Proposal Network (RPN). In the typical Fast R-CNN models, Fast R-CNN utilizes CNN to generate a map of features from the input image. The generated feature map is then progressed to the selective searching procedures, where bounding boxes or region proposals of detected objects are drawn according to features and search results.

In Faster R-CNN, the CNN is still utilized for generating feature maps from images. However, the generation of bounding boxes or region proposals are not the same. Faster R-CNN is introduced with RPN, which is a neural network that shares the same structure as the CNN within. Figure 3.2 shows the framework of Faster R-CNN and relationships of its components [50]. The feature maps extracted from the image is used by two separate parts. The RPN uses the feature maps to predict region proposals which are candidates of bounding boxes for detect objects. The feature maps are also appeared in the RoI pooling part, where they are bounded with the proposals predicted by the RPN. Feature maps and the region proposals are sent to the fully connected layers that include the classifier and the regressor. Classifier recognizes features in the area bounded by the region proposals, while the regressor optimizes sizes of region proposals to fit the object areas.

There are four steps into the training of the Faster R-CNN.

1) The first step is the training of the RPN, in which the RPN is trained individually. The initial weights can be randomized or set with pre-trained networks.

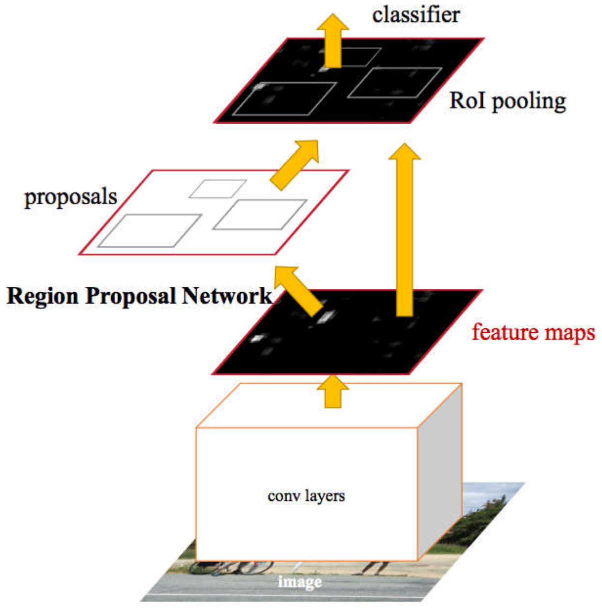

Figure 3.2: Framework of the Faster R-CNN.

2) The Fast R-CNN module for detection is trained individually, using the region proposals from the RPN in the first step.

3) The RPN is re-trained for the fine-tuning purpose. In this step, weights of the RPN and the Fast R-CNN are shared.

4) The Fast R-CNN part is re-trained using the trained RPN from the third step.

Even though there are more advanced object detectors available such as YOLO, the Faster R-CNN is chosen for the landmark detection task instead. Faster R-CNN is chosen due to several reasons. The first reason is that Faster R-CNN is fast enough for real-time implementations on the robots. Secondly, detection accuracy of the Faster R-CNN has no problem with small objects, which may refer to landmarks located in the far distance. Finally, Faster R-CNN require less preprocessing of the raw visual data when used. It reduces programming burdens and computation resources that required for the preprocessing of visual data. Further subsections include the explanations for the preparation of data required in the training of our Faster R-CNN model, and details for the structure of our implemented Faster R-CNN for landmark detection.

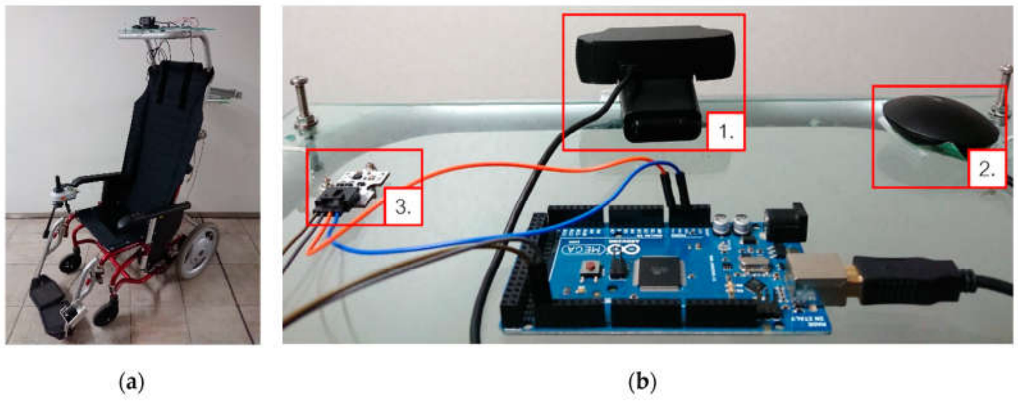

Figure 3.3: Wheelchair robot equipped with sensors for data gathering: (a) overall view of the wheelchair robot; (b) sensors for data gathering: 1. Camera, 2. Global Positioning Systems (GPS) receiver, and 3. Compass sensor.

3.1.1.1 Data Preparation of the Faster R-CNN for Landmark Detection

Faster R-CNN is a supervised deep learning algorithm. It needs to learn representations of features and objects in images. Therefore, the data need to be prepared for the training of the implemented Faster R-CNN model. Since the main goal of the implemented Faster R-CNN in the landmark-based localization system is to detect landmarks in images, visual data with landmarks must be obtained.

The first step into the data preparation is the data gathering, as our desired data is unusual and not available online. The planned dataset includes images and their corresponding geolocation data, which consists of latitude and longitude coordinates and compass orientation. We set the data gathering tools as shown in Figure 3.3. The tools included the wheelchair robot, camera, GPS receiver, and compass sensor. Camera, GPS receiver, and compass sensor were attached to the top of the robot. The wheelchair robot is 55 cm in width, 120 cm in length and 140 cm in height. We used a Logitech C920 HD (Logitech, Lausanne, Switzerland) as the robot camera, BU-353S4 (GlobalSat, Taipei, Taiwan) as the GPS receiver and an Octopus 3-axis digital compass sensor. All images in the dataset were taken from the area near Koganei campus of Hosei University, Japan. Two areas were selected for robot localization in outdoor environments, as shown in Figure 3.4. The length and width of area 1 is 70 and 30 m, respectively. Area 2 is 75 m in

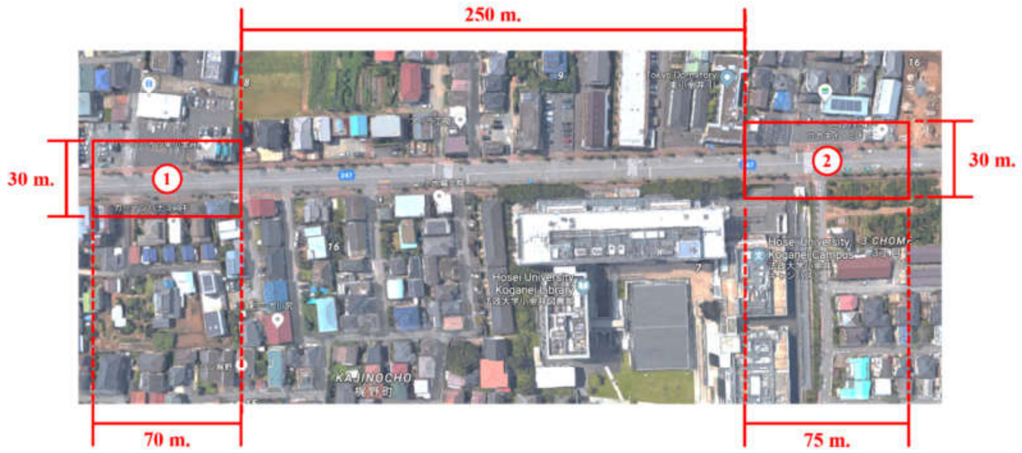

Figure 3.4: Map of the experimental areas (Google map). The areas of experiments are marked with the red rectangles.

length and 30 m wide. The two areas for experiments were in a distance of 250 m from each other. There are different types of landmarks available in each area, which distinguished one experimental area from the another.

The dataset was constructed from 1,625 images in the form of JPEG color images at the size of 320 × 240 pixels. During data gathering, the robot was pushed by a human, and images were taken manually. Each time an image was taken, the corresponding geolocation data was tagged to the image automatically. The tagged geolocation data includes location coordinates and compass orientation. Location coordinates were received from the GPS receiver in the form of a GGA message. Latitude and longitude information inside the GGA message was extracted and tagged to the image. Compass orientation was received from the compass sensor, converted to magnetic compass orientation, before being tagged to the image. We collected the data in different weather conditions in order to increase the robustness of the proposed algorithms. Among 1,625 sets of data gathered, 1,198 sets of data were randomly selected for training, while the remaining sets were used for testing the trained models.

The second step is the labeling. All gathered images in the dataset were hand-labeled with bounding boxes of landmarks in images. Nine types of landmarks were utilized for robot localization: 'FamilyMart', 'CocaCola', 'BicycleLane', 'NoTruck', 'Crossing',

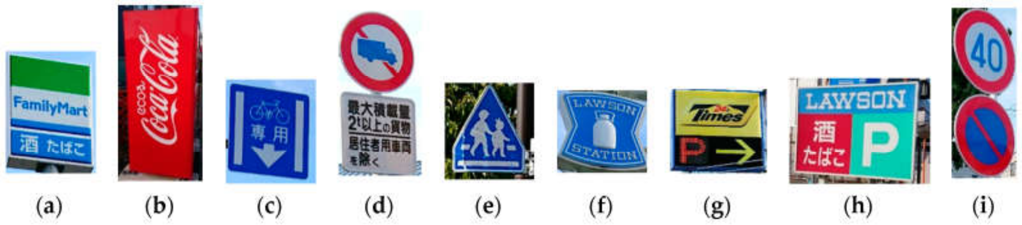

Figure 3.5: Landmarks used in the experiments: (a) ‘FamilyMart’; (b) ‘CocaCola’; (c) ‘BicycleLane’; (d) ‘NoTruck’; (e) ‘Crossing’; (f) ‘Lawson’; (g) ‘TimesParking’; (h) ‘LawsonParking’; (i) ‘RoadSign 1’.

‘Lawson’, ‘TimesParking’, ‘LawsonParking’, and ‘RoadSign 1’. Figure 3.5 shows pictures of these nine landmarks in the area of experiments. Each bounding box is in the form of a vector with four member elements, which contains horizontal and vertical position coordinates of the top-left corner, width, and height of the bounding box in the image. Unit of position coordinates, width, and height of the bounding box is determined by the number of pixels. Horizontal and vertical position coordinates are referenced from top-left corner of the image. For example, a bounding box that has a vector of {10, 20,

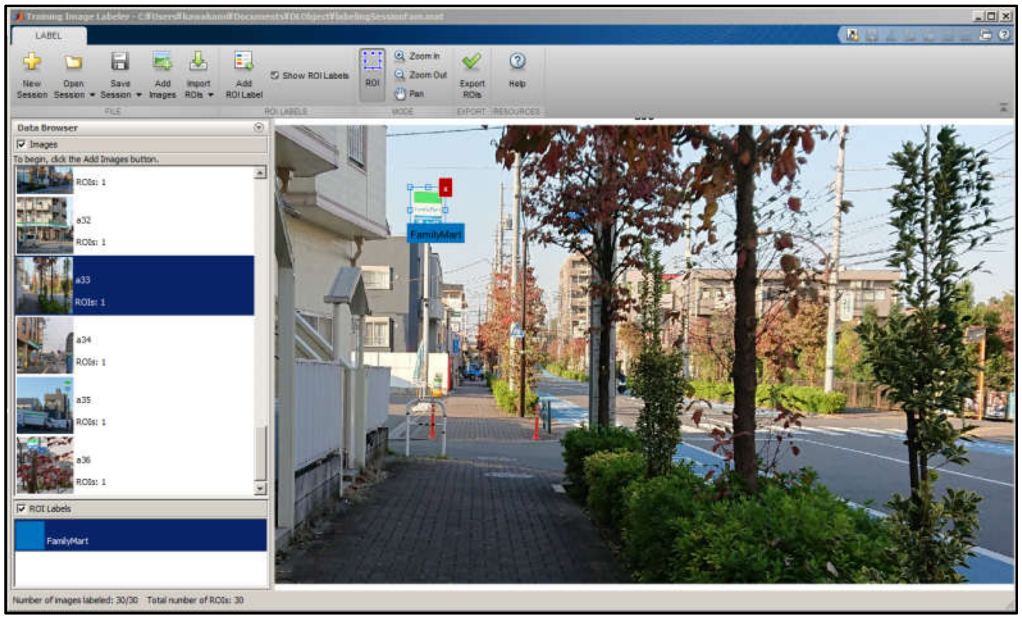

Figure 3.6: Image Labeler application in MATLAB.

56, 72} has its top-left corner at the pixel number 10 horizontally and 20 vertically, and the width and height of the box are 56 and 72 pixels, respectively.

The labeling process was completed in the application named Image Labeler, which is one of applications in the MATLAB software. Sample appearance of the Image Labeler is illustrated in Figure 3.6.

3.1.1.2 Structure of the Faster R-CNN for Landmark Detection

The standard edition of the Faster R-CNN was implemented for the landmark detection task. Figure 3.7 shows the structure of the implemented Faster R-CNN model. The process of our Faster R-CNN is the same as the model proposed in [50]. The difference between out Faster R-CNN and the vanilla version is the structure of the CNN inside. Our CNN in the Faster R-CNN were designed based on trial-and-error method and different principles for designing CNN in [132].

The set of convolutional layers of the CNN analyzes the whole input image to construct a convolutional feature map. As the size of the smallest landmarks in the utilized dataset is nearly 32 × 32 pixels, the input size is set to 32 × 32 × 3, where the last 3 is for

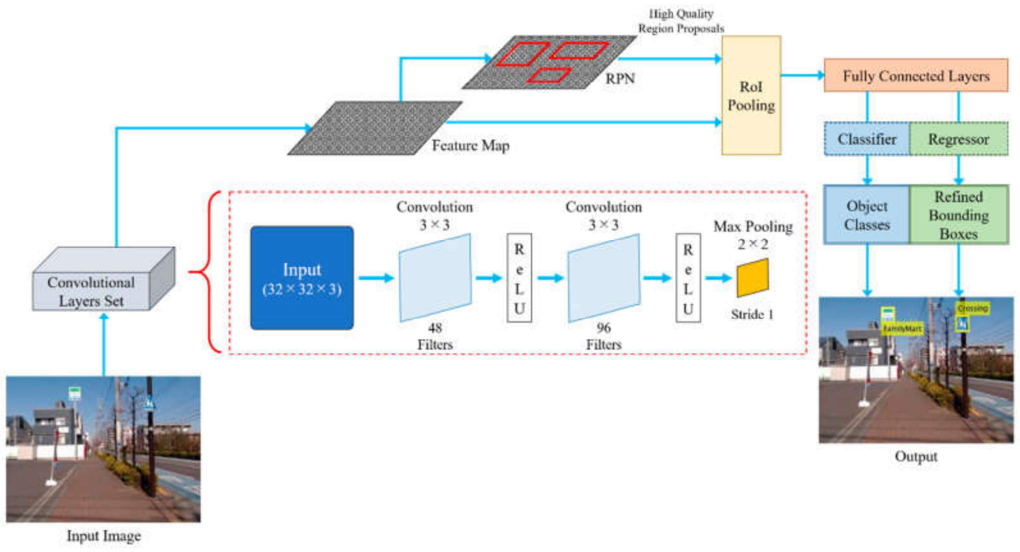

Figure 3.7: Faster R-CNN for landmark detection and its components.

three color channels: red, green and blue. The set of convolutional layers contains two, two-dimensional convolutional layers, with a rectified linear unit (ReLU) attached after each convolutional layer. The set also includes one max pooling layer for down-sampling purposes. Each convolutional layer employs a 3×3 filter and has the stride settings of 1 pixel for both horizontal and vertical strides. The number of filters in the first convolutional layer is 48, while 96 filters are used for the second convolutional layer. The max pooling layer is placed at the end of the layers set, in which the pooling size is 2×2 and the stride settings is 1 pixel for both horizontal and vertical strides. This small pooling size is applied to prevent premature down-sampling of the input image, which may cause the loss of features in the result feature map. The training of our Faster R-CNN uses the whole images as input, and the labeled bounding boxes as the target. Training continues for 20 epochs, with 1×10^{-4} initial learning rate.

3.1.2 Feed-Forward Neural Network for Location Estimation

Feed-Forward Neural Network (FFNN) is an artificial neural network which has its internal connections in such ways that do not form a cycle. FFNN is the simplest artificial neural networks available. It contains nodes, layers, and connections between nodes, which resembles the way neural systems in animals work. FFNNs usually contain only three layers, consisting of the input layer, the hidden layer, and the output layer. Input layer of FFNNs acquire inputs to be processed in the network, while the computations are done in the hidden layer. The answers are given in the output layer, according to the representations of input data. Despite its simplicity, FFNNs possess the ability of learning representations of data and desirable outcomes from the given data, same as advanced deep learning algorithms. The learning or the training processes of FFNN usually include back propagation algorithms which help the networks better understand representations of data that they must work with.

The localization part of the landmark-based localization system has the goal of generating robot locations from bounding boxes and labels of detected landmarks from the Faster R-CNN. Due to the simplicity and the ability of learning the representations of data, FFNN is chosen for the localization part. The FFNN for localization considered the bounding boxes which arranged by their labels as the input. From labeled bounding boxes, FFNN must generate proper latitude and longitude coordinates, along with proper compass orientations. Further details on the data preparation for the FFNN and the implemented FFNN structure are described in following subsections.

3.1.2.1 Data Preparation of the FFNN for Localization

Same as all other neural networks, FFNN requires data to train its models. With proper settings and training data, FFNN can understand input data in its given tasks and give proper answers regarding the input. In this subsection, the matters of data preparation for the training of the FFNN for localization are discussed.

The main goal of the localization part is to generate proper latitude and longitude coordinates and compass orientations from labeled bounding boxes of detected landmarks. The first step in the data preparation is data gathering. The data for the FFNN were shared with the Faster R-CNN part. As mentioned in 3.1.1.1, the gathered data were in the form of images that embedded with location coordinates from the GPS receiver and compass orientations from the magnetic compass. The total number of gathered data are 1,625 sets. The gathered images were labeled through the application called Image Labeler which resides within MATLAB software. The labeled images include labeled bounding boxes which can be used to train both the Faster R-CNN and the FFNN.

In the training of FFNN, labeled bounding boxes of landmarks from the Image Labeler are arranged by their labels. There are nine classes of landmarks for the localization system, as mentioned in 3.1.1.1. Each class can store two bounding boxes of four elements at most. Placeholders of bounding boxes for each landmark class are filled with {0, 0, 0 ,0} in the case that the bounding boxes are not available. Among 1,625 sets of data gathered, 1,198 sets of data were randomly selected for training. These random sets of data were the same sets which are used to train the Faster R-CNN.

3.1.2.1 Structure of the FFNN for Localization

The implemented structure for the FFNN for localization is illustrated in Figure 3.8. Our FFNN structure consists of 72 input neurons, 48 neurons in the hidden layer, and 3 neurons for the output layer. The number of 72 for the input neurons comes from the input

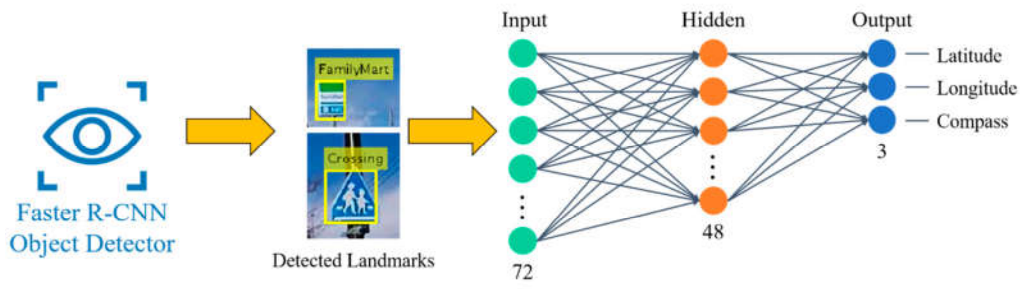

Figure 3.8: Feedforward neural network (FFNN) for localization with detected landmarks from Faster R-CNN.

that contains arranged placeholders of bounding boxes from nine landmark classes, in which each class having two placeholders for bounding boxes of detected landmarks. Since each bounding box has four elements for its position in the image and its size, 18 placeholders need to be multiplied by 4, having 72 input neurons for each bounding box element as the result. The output neurons of 3 come from 3 desired outputs, including values of latitude, longitude, and compass orientation. The number of hidden neurons of 48 is found through optimization in the trial-and-error process.

3.2 Vision-Based Navigation System with Deep Q Network

Navigation systems require decent decision makers to guide the robots to their goals. The second system proposed in this thesis is the vision-based navigation system for outdoor mobile robots using the Deep Q Network (DQN) [133]. DQN is the significant milestone in the field of deep reinforcement learning. It combines CNN and Q-Learning together. As based on Q-Learning, DQN is an off-policy algorithm. It finds the optimal policies that actions of agents under such policies maximize total cumulative rewards from the consequences of actions to the environments. DQN replaces tables in Q-Learning with neural networks. The networks in DQN generates action values for the DQN agents instead of looking up the action values in the table. The strategy is still the same as Q-Learning though, in which actions with the highest action values in any given states are chosen. Training strategies in DQN are introduced with the experience replay and the target-action value function. Experience replay is the strategy in which experiences of actions are stored in the memory, and randomly selected to train the DQN model. Experiences generally contain the state, action committed, reward received from such action, and the consequence state after the action. States in experiences are originally stacks of visual frames from video games [87]. The calculation of target action-values can be completed through two equations:

$$y_i = r_i \tag{1}$$

$$y_i = r_i + g \times max_{a'} Q_n (s_{i+1}, a') \tag{2}$$

where y_i is the target-action value of the experience i, r_i is the reward from committing an action stored in the experience, g is the reward discount, and $max_{a'} Q_n (s_{i+1}, a')$ is the maximum target-value from the target-value function in the consequence state of the experience. Target action-value will be calculated by (1) if the training episode of DQN is terminated at the given step. Calculation from (2) will be applied to other cases.

We implemented the vanilla version of the DQN in our navigation system. The training process of DQN agents includes the exploration and exploitation, in which the agents find optimal solutions through the trial-and-error strategy. This process can harm parts of the robots if the training process is carried on the real robot directly. The simulation environments based on the First-Person Shooter (FPS) game was created for the training of DQN agents in navigation tasks. ViZDoom platform [134] was chosen for creating the simulations for our DQN agents. The game-based platform was chosen over traditional simulations platforms because it contains convenient resources, which can be used to create simulation environments faster and simpler. However, the choice of creating the game-based simulations led to another problem. There are big differences between simulations and the real world. As discussed in chapter 2, many works transfer experiences from the simulations to the real world by applying transfer learning strategies. We saw these strategies to be heavy and require a large amount of efforts and resources. Instead, a marker-based Augmented Reality (AR) algorithm was employed to calibrate visual data from the real camera to be similar with experiences in the simulation.

It resulted in the navigation system illustrated in Figure 3.9. The simple marker-based AR algorithm and a simple color-based object detector are used to enhance robot

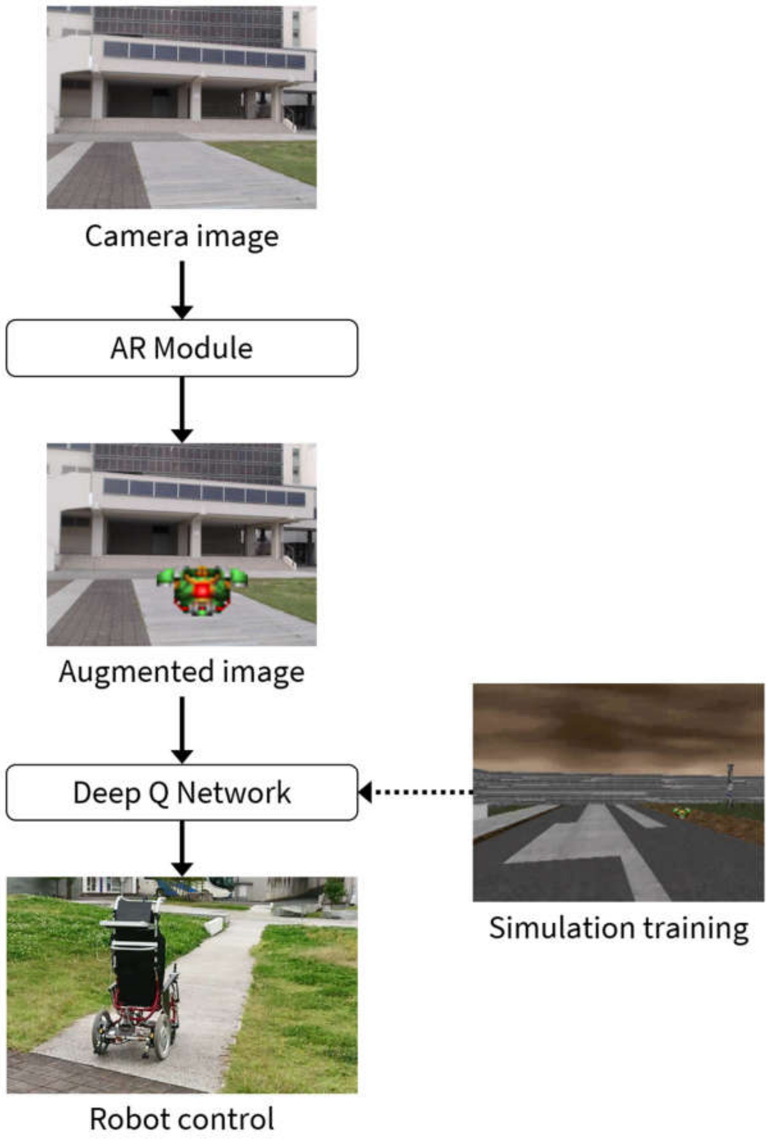

Figure 3.9: Diagram of the outdoor robot navigation system with DQN.

perspectives. Camera image is fed to the AR module, where the goal object in the image is detected. The goal is replaced with the simulated object that DQN was trained with. Therefore, our DQN decides proper actions for the robot from augmented camera images. There are three main components in our navigation system, including DQN, simulation environments, and the AR module. Further details of the navigation system components are described in following subsections.

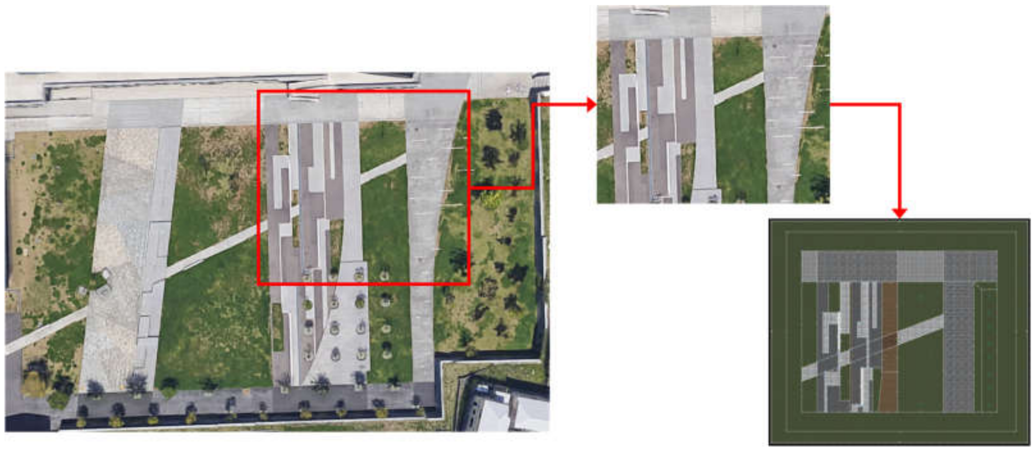
Figure 3.10: The area used for the simulation.

3.2.1 Simulation Environments based on ViZDoom Platform

ViZDoom is a software platform for machine learning research based on the FPS game Doom which was released in 1993 [134]. It contains convenient game resources and easily accessible sets of codes that link between the simulation and control programs. Simulation environments in ViZDoom can be created faster and easier than building simulations from their bases. Doom game itself is lightweight and can be processed fast on typical personal computers. ViZDoom is also compatible with different game editing tools and is highly customizable. The simulation environments created for ViZDoom are projects to their agents in first-person perspective. This first-person perspective is the same perspective that mobile robots can see through the attached camera, supposed that the camera is directed to the front of the robots.

Doom Builder [135] was the tool selected for creating simulation environments for our DQN agents. The simulation environment was created with the field of Hosei University, Koganei campus, Japan, as the reference, as illustrated in Figure 3.10. Details of the created environment were not precisely measured. Textures and objects in the environments were also of the video game resources. Settings of the simulation can be completed via different methods. Game settings such as the spawn locations, background reactions in the environments, and game completion criteria, can be set through the scripts in Action Code Script (ACS) language. The ACS scripts are embedded with the game

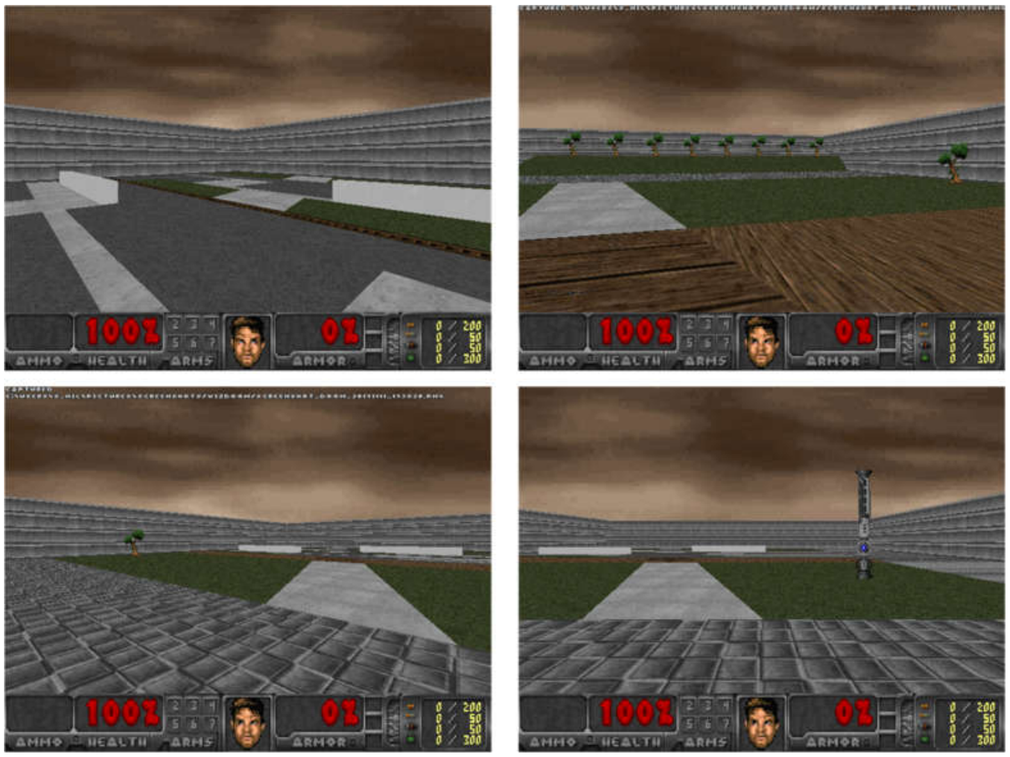
Figure 3.11: Sample of the created simulation environment.

environments created by Doom Builder and can be edited with the Doom Builder. Simulation settings such as simulation time, rewards, and possible actions of agents, can be set with an additional text file. Figure 3.11 illustrates some samples of the created environment in the first-person perspective. Figure 3.12 compares the real world and the created simulation environment. Interactions between the environment and artificial agents are more oriented into games. Agents in the simulation can interacts by moving and gathering items. The simulation can respond to the agents in many ways, such as increasing or decreasing agent status, giving rewards, and the agent termination.

Simulation settings of the created simulation environment are shown in Table 1. The step limit is set to 250 steps in each simulation instance, in which one step is set to 1/35 s. This means one simulation instance has the time limit of approximately 7 s. Visual data that agents see from the simulation is in the form of RGB frames at the size of 320×240 pixels. Agents can commit 3 actions, including moving forward, turning left, and turning right. Rewards from the environment are stated in Table 2. Agents tend to move toward

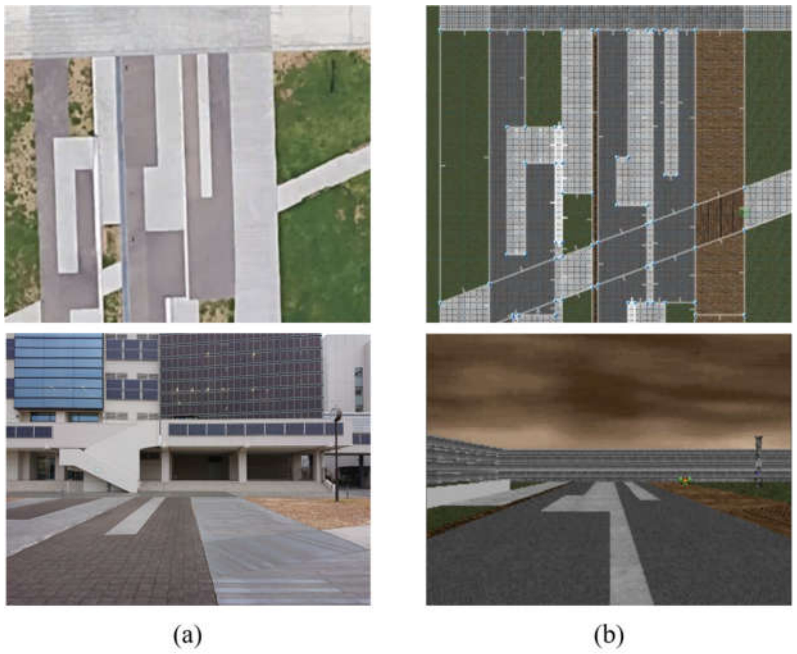

Figure 3.12: Comparisons between (a) the real world and (b) the simulation.

the goal as fast as possible, since in each step -3 is added to the accumulated reward. Distances between the agent and the goal are also calculated as rewards, which become more negative as the agent moves away from the goal. Termination state is also considered bad. The large negative reward of -2,000 will be given if the agent moves into the grass and terminated. On the other hand, the rewards are increased as the agent moves toward the goal, with the big reward of +2,000 provided as the agent reaches the goal. A simulation instance has three termination conditions: (1) the agent reaches the goal, (2) the agent is terminated, and (3) the simulation exceeds the step limit.

Table 1. Simulation settings of the created environment.

Simulation properties	**Applied settings**
Step limit per run	250 steps
Time per step	1/35 s
Agent input	320×240×3 (RGB image)
Agent actions	3 (forward, turn left, turn right)

Table 2. Reward provided by the simulation environment.

Agent states	Provided rewards
Moved	– (distance_x + distance_y) – 3
Dead (step on grass)	– 2,000
Goal	+ 2,000

The task for the DQN agent for navigation is the simple navigation task, in which the starting location is fixed, and the goal location is random within the specified area. The simulation area is limited to approximately 20×18 m, as seen in Figure 3.13. This simulation area was used for both DQN training and experiments. The robot starting location and orientation are fixed, so as the target area. The target area is 5 m far from the robot starting position. The robot initially faces toward the target goal location. The target area has the size of 10×5 m, with the goal object randomly generated inside it. Green grass areas are set to instantly terminate the agents if stepped on it.

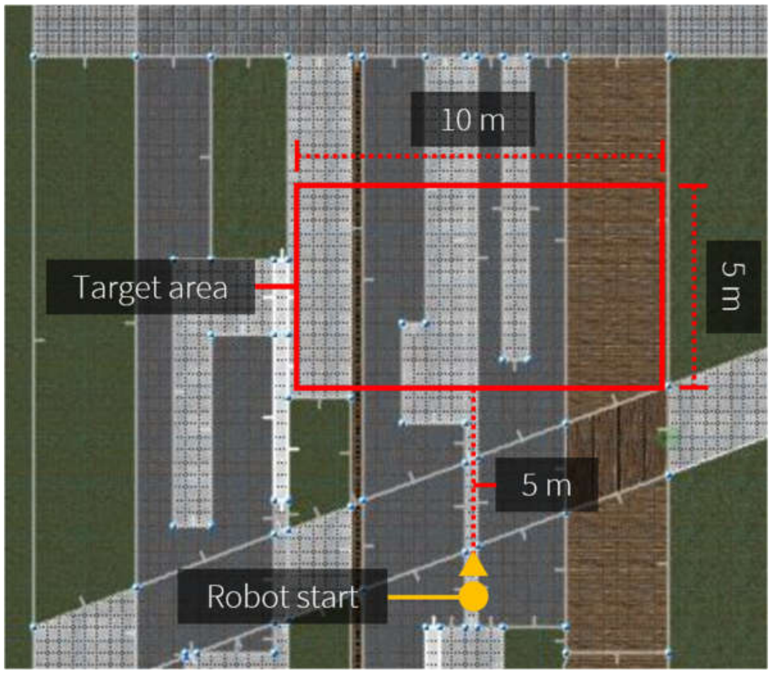

Figure 3.13: Simulation area with the agent (robot) start and the target area.

3.2.2 Deep Q Network for Vision-Based Navigation

The vanilla version of the DQN is employed for the navigation task. Settings and learning process are the same as the original proposal in [87]. The differences in our DQN include the structure, training settings, and the way our states is stored. Structure of our DQN is displayed in Figure 3.14. Visual data is fed through the input layer, continuing into convolutional layers, then processed through a fully connected layer and output one

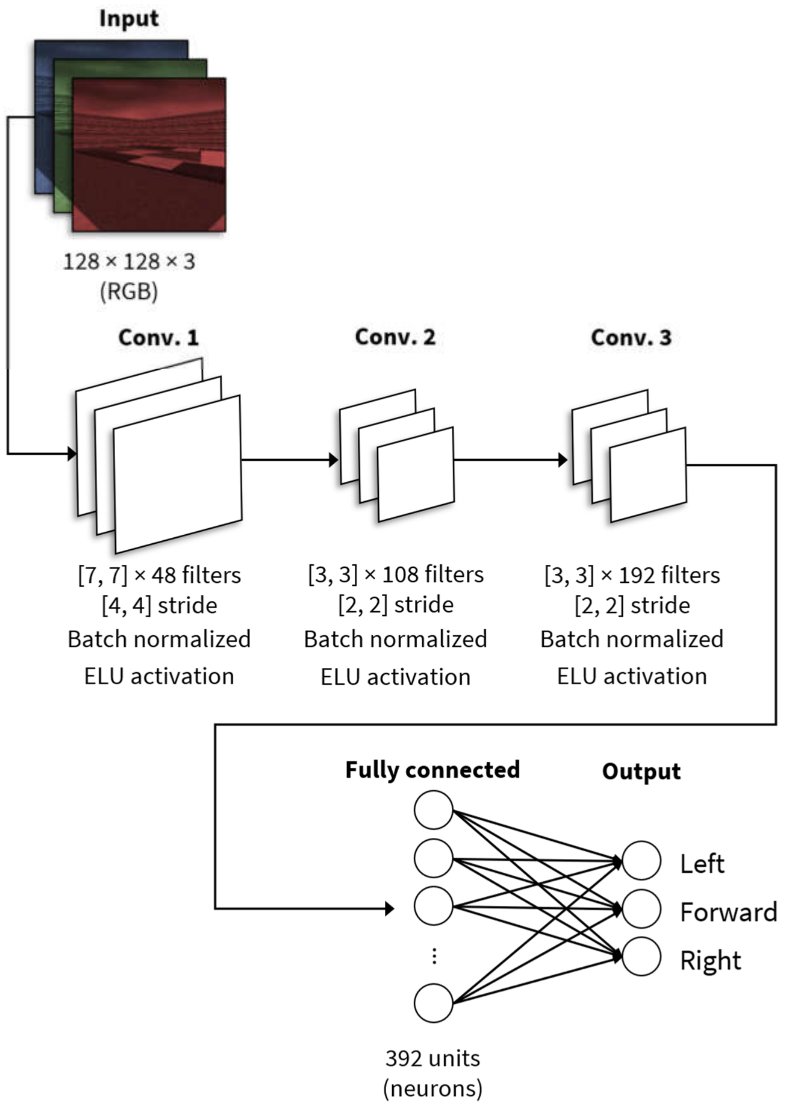

Figure 3.14: Structure of the CNN in DQN for navigation.

Table 3. DQN training settings.

Training properties	Applied settings
Episodes	1201
Steps per episode	250
Initial exploration parameter	1.0
Exploration decay	0.000192
Learning rate	0.00024
Reward discount	0.949
Memory batch per training	96
Memory size	1,000,000

of three possible actions. The input size of the DQN is 128×128×3, which is refers to the size of RGB image. Our DQN considers a single RGB image as an input, instead of a traditional stack of luminance frames [87]. The single RGB image was employed to reduce effects of framerate differences between the real robot camera and the simulation. However, the use of single RGB image input for DQN can cause temporal limitation problems and reduce DQN performances. Convolutional layers in our DQN were inspired by the pyramidal design [136]. The first convolutional layer has a large filter size of 7×7, with the stride of 4×4. The later convolutional layers apply small filters of 3×3, with stride 2×2. The number of filters starts at 48 for the first convolutional layer. For the deeper convolutional layers, the number of filters has increased to 108 and 192 in the second and the third convolutional layers, respectively. Batch normalization and ELU activation are attached after each convolutional layer. Our DQN structure was optimized through trial-and-error process.

During training and navigation, the input RGB images have the first 20 rows on the top of images cropped. The removed areas mostly are the sky or the background buildings, which have no impact on the robot navigation. The cropped images are resized to 128×128×3 pixels, then fed to the DQN. As aforementioned, we implemented the vanilla DQN and its training process for our DQN. Variables for our DQN training are different from [87], as described in Table 3. The training variables were set through trial-and-error procedures. We used the "Basic" scenario in ViZDoom [134] to test our DQN structures and training settings, before tuning the DQN to our navigation simulation. The number of episodes was set to 1201, with the step limit of 250 steps per episode. The epsilon-greedy algorithm was implemented with the exploration parameter of 1.0 and the exploration decay rate of 0.000192 for each performed action step. Learning rate of the network was set to 0.00024. The rewards for calculating the target-action values were discounted by the factor of 0.949. The DQN was trained in every step of actions with experiences from the memory, in which the experiences were sampled in a batch of 96. The memory size was set to be able to store 1,000,000 experiences at maximum.

Each value in the settings has significant meaning to the DQN training. The exploration factor and exploration decay play important roles in balancing the exploration and exploitation process. The reward discount determines probabilities of the end of episodes. More reward discount makes the DQN agents consider more on long-term rewards. Episode rewards during training are convolved and shown in Figure 3.15. The DQN robot agent chose proper actions and received more rewards in later episodes.

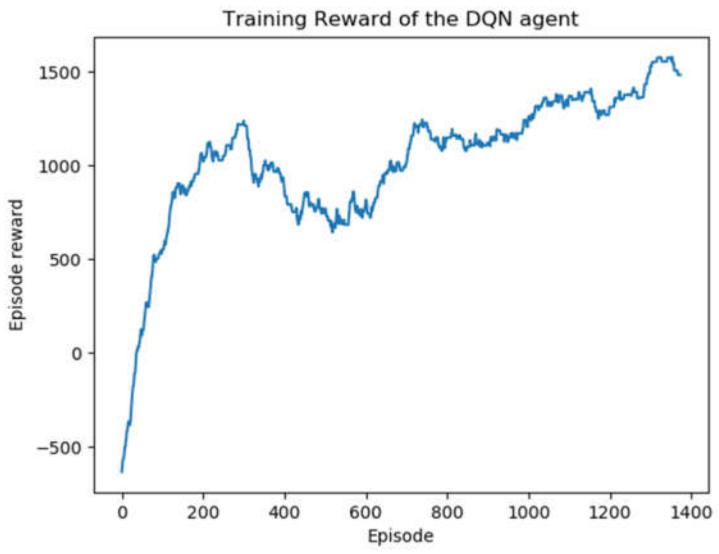

Figure 3.15: Training reward of the DQN agent for navigation.

3.2.3 Marker-Based Augmented Reality

Augmented Reality (AR) algorithms are algorithms for augmented visual information received from the reality. They can be categorized into different sectors. By markers, AR algorithms can be divided into the marker-based AR algorithms and marker-less AR algorithms. Marker-based AR is considered to be the basic method of implementing AR, since it typically requires only markers and some computer vision algorithms [137]. On the other hand, marker-less AR require different types of sensors to augment visual information, such as the GPS and compass.

The marker-based AR algorithm is employed to improve the robot vision in our navigation system. A simple color-based detector was used in our AR to detect the marker

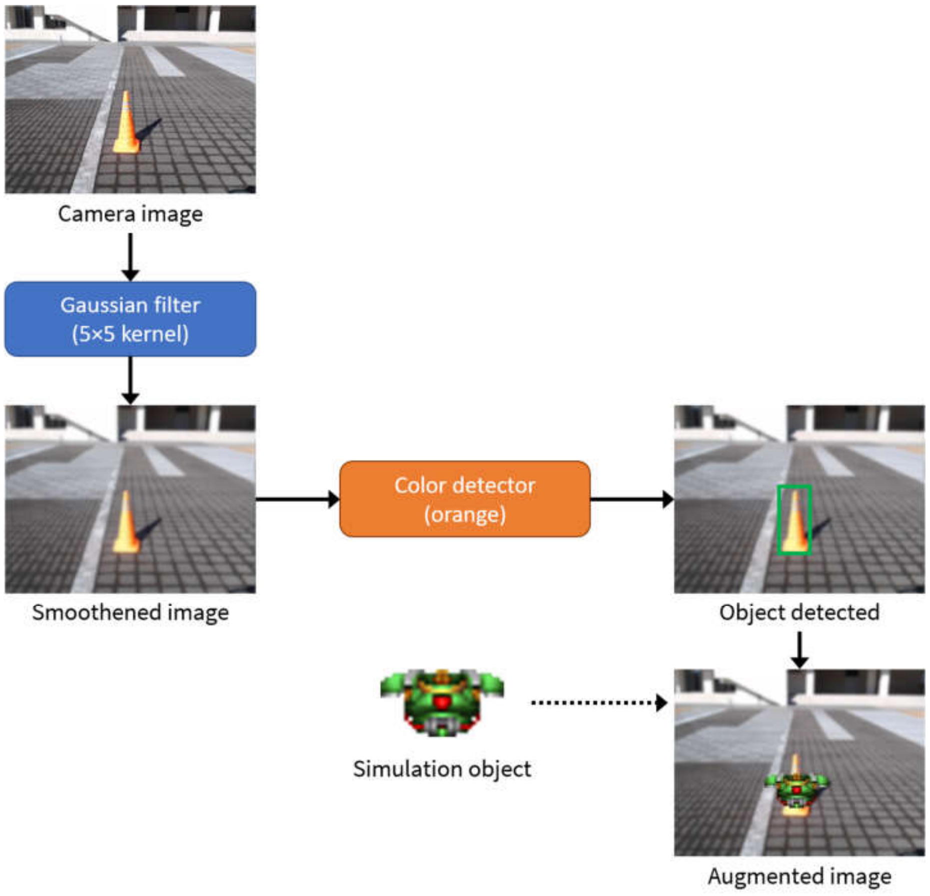

Figure 3.16: Flows of the augmented reality module.

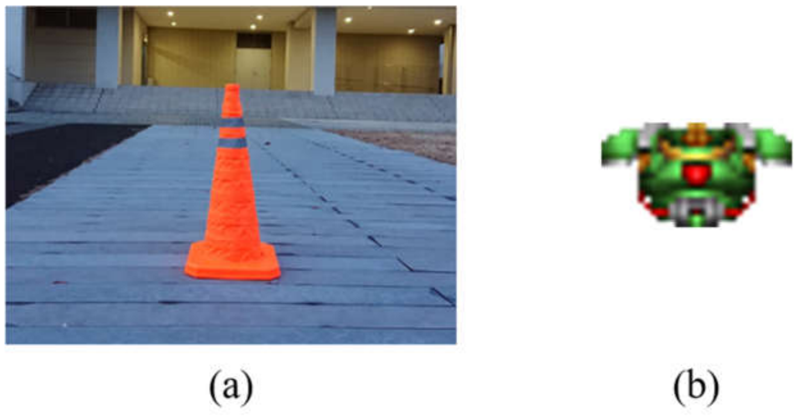

Figure 3.17: The goal objects: (a) the real traffic cone, (b) the green armor.

objects in camera images. The color-based object detector was selected over neural network-based detection algorithms due to its lightweight and simplicity. Neural network and deep learning detection methods such as the method in [138] require the detectors to be trained. Neural network-based methods also consume more computation resources than simple color detectors when used. Flows of the AR module is illustrated in Figure 3.16. We selected the goal object which has different color to the environment. Camera image is smoothened through the filter to reduce noises. We used a gaussian filter with the kernel size of 5×5 to smoothen camera images. The object detector then finds the goal by detecting object colors. The goal object is detected and replaced with the object from the simulation, which resized according to the goal object.

Since an object is needed to be detected as the real goal, we used a traffic cone as the goal object. The traffic cone was chosen because of its appearances. The appearances of traffic cones are the same from all directions. Colors of the traffic cones are usually unique and recognizable in almost every environmental condition. Its orange color can be easily detected with specific hue values. Our AR module replaces the traffic cone in Fig. 3.17 (a) with the green armor from the simulation in Fig. 3.17 (b). Fig. 3.18 shows the traffic cone in camera images and their augmented images at different ranges. We augmented camera images by replacing the traffic cone with the green armor, before feeding the augmented image to the DQN for robot navigation. The replaced object was smaller as the distance increased.

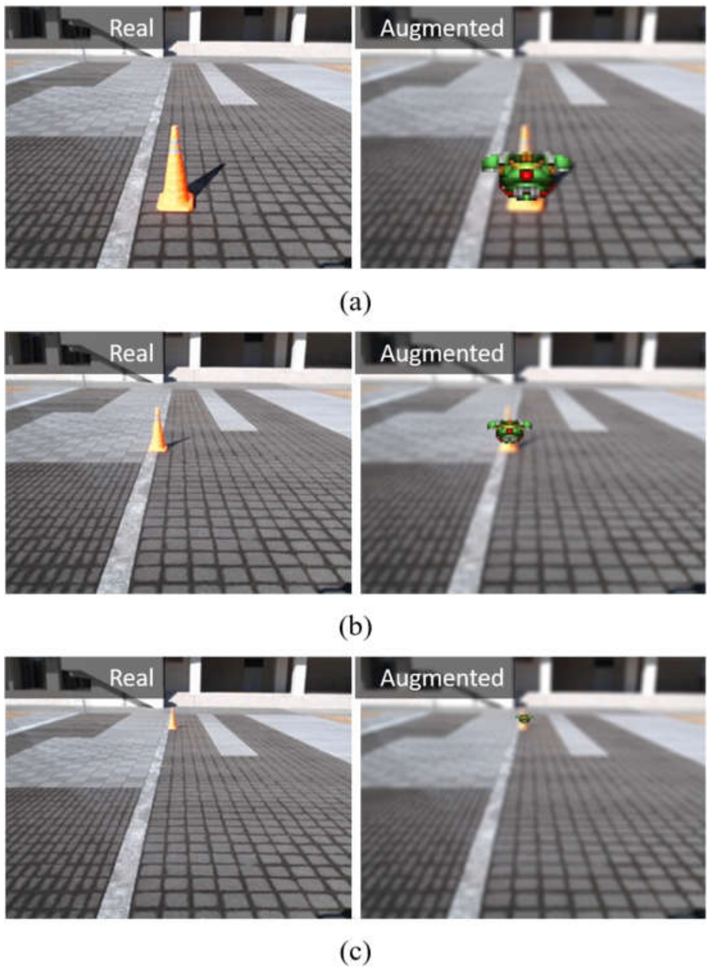

Figure 3.18: Camera images (left) and augmented images (right) at (a) 2 m, (b) 5 m, and (c) 10 m.

3.3 Improvement of Deep Q Network for Outdoor Navigation using the Transfer Learning Strategy

The DQN can be used to navigate mobile robots with visual information from the attached camera. More navigation tasks added to the robot result in more DQN agents that need to be trained for each specified task. This repeating process of acquiring new tasks and training new DQN agents greatly consume computational time and resources. As discussed in chapter 2, transfer learning strategies can be used to transfer knowledge from the source task to others.

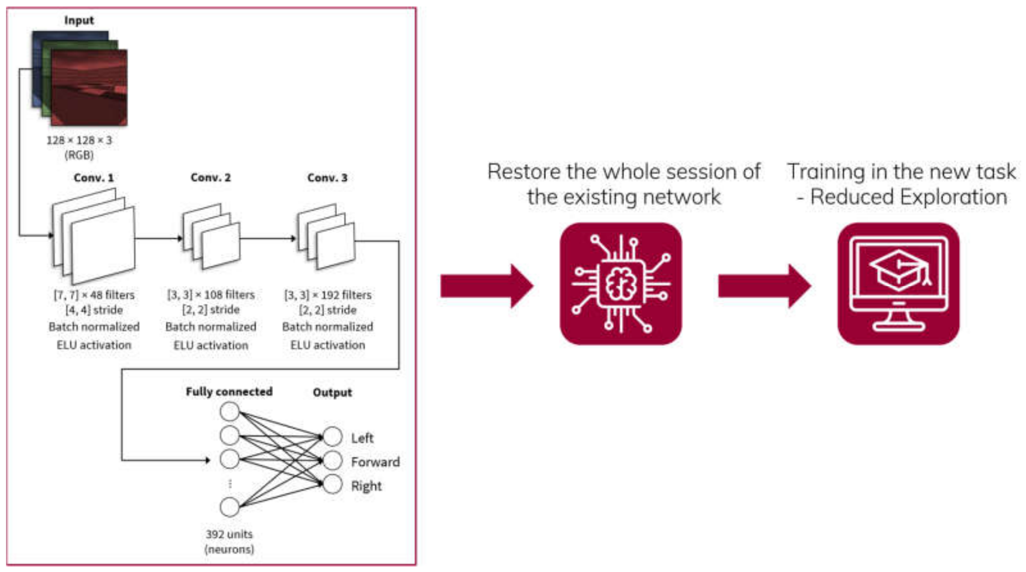
Input
128 × 128 × 3 (RGB)
Conv. 1
Conv. 2
Conv. 3
[7, 7] × 48 filters [4, 4] stride Batch normalized ELU activation
[3, 3] × 108 filters [2, 2] stride Batch normalized ELU activation
[3, 3] × 192 filters [2, 2] stride Batch normalized ELU activation
Fully connected
Output
Left
Forward
Right
392 units (neurons)
Restore the whole session of the existing network
Training in the new task - Reduced Exploration

Figure 3.19: Simple diagram of the employed transfer learning strategy.

A transfer strategy is employed to transfer knowledge from the simple navigation task in section 3.2 to other tasks. From the simple diagram in Figure 3.19, the whole session of the trained DQN agent is loaded for the training of the new model without changes. In order to utilize the knowledge in the loaded DQN agent, the initial exploration factor is reduced. The training of the new agent has less chance to select random actions, and select actions based on prior knowledge instead. This process helps the DQN agent to find the optimal solution faster than fully randomized trial-and-error strategy.

CHAPTER 4

EXPERIMENTAL RESULTS

Three deep learning-based systems have been proposed for the localization and navigation tasks of mobile robots. Each system has its own task and specialized characteristics. The methods of how each system works were described in the third chapter of this thesis.

In this chapter, three proposed systems were tested with different experiments according to their assignments. The first system of localization was tested with the experiments for object detection and the localization tests. The second system for navigation was tested with experiments in the simulation and the real robot. The third system was tested with simulation settings that contain higher difficulties than the second one. Experimental results and the robot which was the testbed for two of the proposed systems are described in following sections.

4.1 Wheelchair Robot Platform

The wheelchair robot is the testbed for two of the proposed deep learning-based systems. Appearance of the wheelchair robot is illustrated in Figure 4.1. The wheelchair robot itself consist of two Yamaha AC motors and a motor controller. The AC motors are powered by a 24-voltage battery which can be stored beneath the seat. Motors can be controlled through serial commands. Users can send commands from the laptop that can be attached to the motor controller via a USB cable. Commands for controlling the motors are only for controlling speed and steer of the motors.

The wheelchair robot has three visible compartments, located on the top, back, and the lower left side of the robot. The top compartment can be placed with different sensors

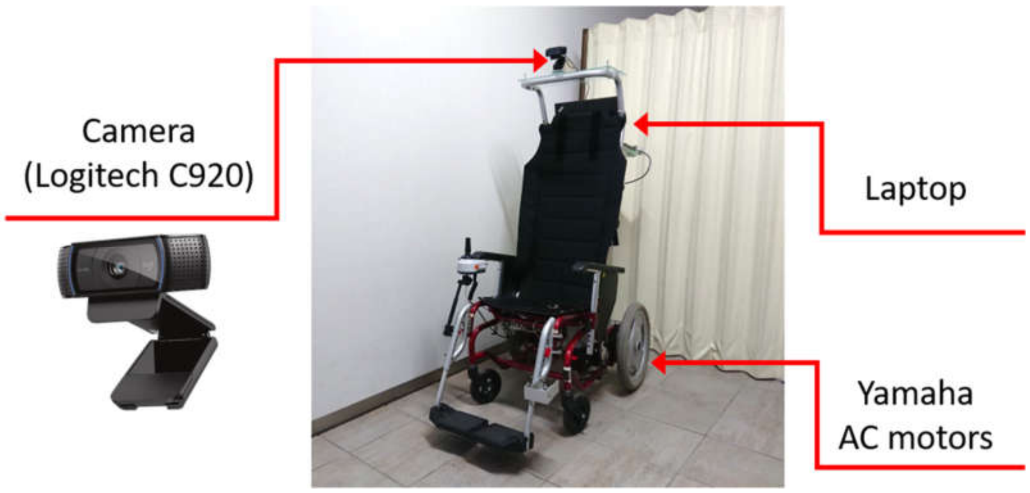

Figure 4.1: Wheelchair robot platform.

such as GPS receiver and magnetic compass. In our experiments however, it was placed with only a single web camera. The back compartment can store a laptop which is used as the main controller. The lower left compartment is designed specifically for housing a small laser rangefinder module.

4.2 Landmark-Based Outdoor Localization System

The localization system contains the detection part by Faster R-CNN and the localization part using FFNN. We conducted different experiments according to capabilities of each module in the localization system. There are detection tests for the Faster R-CNN detector and localization tests of the FFNN included in this section.

4.2.1 Detection Test of Faster R-CNN for Landmark Detection

The goal of the landmark detection experiments was to evaluate the performance of the Faster R-CNN, since the localization part is strongly related with the landmark detection. All 427 images remaining in the test set were processed through the Faster R-CNN and embedded with bounding boxes and labels of landmarks detected by Faster R-CNN. Evaluation of detection results includes the qualitative and quantitative tests.

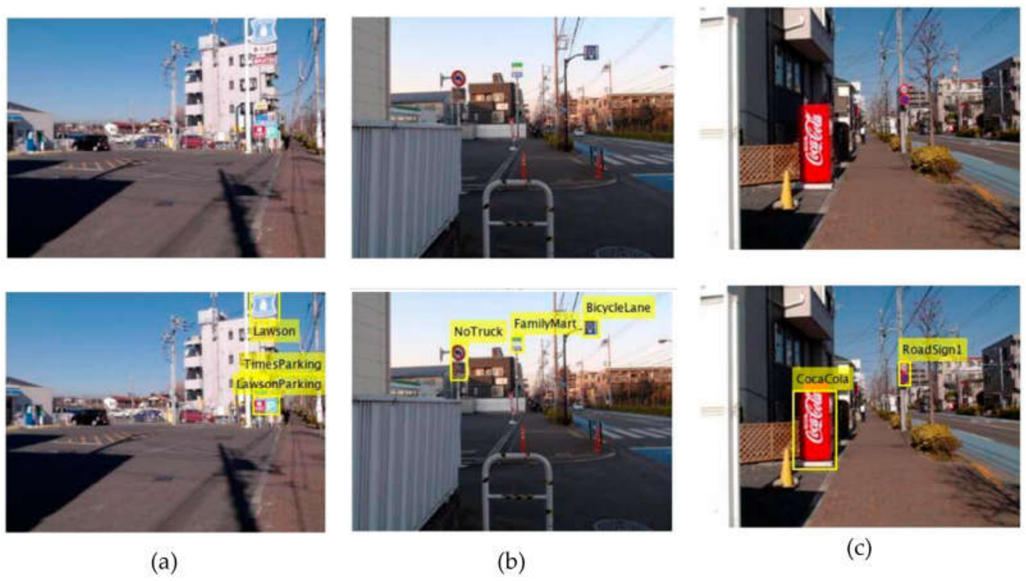

Figure 4.2: Samples of images in the test set and the Faster R-CNN landmark detection results: (a) Sample 1; (b) Sample 2; (c) Sample 3.

The qualitative evaluation was done by analyzing the detection results through human eyes. Some of detection results from the Faster R-CNN on images in the test set are shown in Figure 4.2. Most of generated bounding boxes are placed well on detected landmarks with proper positions and sizes. Labels attached to the boxes correspond to the classes of landmarks shown in Figure 3.5. However, some landmarks such as 'CocaCola' in Figure 4.2 (c) has its bounding box placed in the area of the actual landmark, but the box size did not match with the landmark size.

For the quantitative evaluation, Mean Average Precision (mAP) was used for the quantitative evaluation in landmark detection experiments. mAP is considered to be the actual metric to measure the accuracy of object detectors. The mAP is the mean value of average precisions (AP) from all object classes. In this paper, we refer to this as landmark classes. AP is the average of maximum precisions at different recall values, in which both precision and recall can be calculated by the following equations:

$$P = \frac{TP}{TP+FP} \tag{3}$$

$$R = \frac{TP}{TP+FN} \tag{4}$$

where *P* is the precision, *R* is the recall, *TP* is the amount of the correct bounding boxes comparing from detection and reference boxes in the dataset, *FP* is the amount of missed or misplaced bounding boxes that appeared in detection results, and *FN* is the amount of missed bounding boxes that did not appear in detection results, but existed in the reference dataset. The correct bounding boxes were measured from the ratio of Intersection over Union (IoU), which is the ratio between the intersection area and union area of bounding boxes, comparing detection results with reference data. The higher IoU ratio means less detection error allowance, which can also reduce the outcome of AP values. In this paper, the IoU of 0.5 and 0.7 were employed for measuring detection accuracy, similar to [50] which used an IoU of 0.7. AP values of all landmark classes and the mean values (mAP) of 0.5 and 0.7 IoU ratio values are displayed in Table 4.

Table 4. AP of detection results from the Faster R-CNN.

Class	**$AP_{0.5}$**	**$AP_{0.7}$**
1 (‘FamilyMart’)	0.9024	0.8786
2 (‘CocaCola’)	0.8281	0.5823
3 (‘BicycleLane’)	0.8040	0.5466
4 (‘NoTruck’)	0.8573	0.8573
5 (‘Crossing’)	0.8500	0.8500
6 (‘Lawson’)	0.7682	0.7206
7 (‘TimesParking’)	0.6156	0.4966
8 (‘LawsonParking’)	0.8360	0.7815
9 (‘RoadSign1’)	0.9904	0.9235
Mean	**0.8280**	**0.7375**

From Table 4, the mAP values are 0.8280 and 0.7375 for 0.5 and 0.7 IoU, respectively. This implies that the landmark detection accuracies of Faster R-CNN are 82.80% for 0.5

IoU and 73.75% for 0.7 IoU. Though mAP values were higher than 80% when IoU is 0.5, mAP decreased to around 70% as IoU increased to 0.7. This means landmarks could be detected but may not be precise or have high accuracy. This reduction in detection accuracy is the cause of a lower localization accuracy, as landmark detection results are required to generate localization results in the Faster R-CNN localization method.

4.2.2 Localization Experiments

The localization methods presented in this paper were implemented and evaluated in several localization experiments. All 427 images in the test set were processed in the Faster R-CNN localization system to generate localization results. We added a CNN localization system based on the well-known CNN for classification called 'AlexNet' [139] to compare its performances with our Faster R-CNN. We replaced the last layers of AlexNet with regression layers, which cause the AlexNet to be able to generate localization results from the whole images. The AlexNet was trained with the same training dataset as the Faster R-CNN system. Results from localization methods, including location coordinates in latitude and longitude, and compass orientations were then passed on to the evaluation.

Evaluation of localization results was done by calculating absolute errors and the distance between two points, the generated results and the reference geolocation data in the test dataset. Three absolute errors were considered in the experiments: mean, minimum and maximum absolute errors. The mean absolute error is calculated from the following equation:

$$MAE = \frac{1}{n} \times \sum_{i=1}^{n} \left| a_i - b_i \right| \tag{5}$$

where *MAE* is the mean absolute error, *a* is the result from localization, *b* is the reference value in the test dataset, and *n* is the amount of data in the test set, which was 427 in the experiments. The minimum and maximum absolute errors are the smallest and largest values in absolute errors.

The distances between two points were calculated from the location coordinates of the generated and reference data. We used the haversine formula to calculate distances from latitude and longitude of two points. The haversine formula is widely used in computer programming to determine the distance between two points on a great sphere, which commonly referred to as the Earth. The implemented haversine formula is as follows:

$$D = 2 \times r \times \arcsin\left(\sqrt{\sin^2\left(\frac{y_2 - y_1}{2}\right) + \cos(y_1) \times \cos(y_2) \times \sin^2\left(\frac{x_2 - x_1}{2}\right)}\right) \quad (6)$$

where D is the distance between two points in kilometers, r is the earth radius, which were applied as 6378.1 km [140], y_1 is latitude of localization results in radius, y_2 is the reference latitude in radius, x_1 is longitude of localization results in radius, and x_2 is the reference longitude in radius.

In addition to absolute errors and distance errors, we also calculated the standard errors from localization results and distance errors. The standard errors were calculated to measure deviations of all results, in which the equation for standard errors can be described mathematically as;

$$SE = \frac{\sigma}{\sqrt{n}} \quad (7)$$

where SE is the standard error, σ is the standard deviation of the result, and n is the amount of data, which was 427 for the test set.

Table 5 shows each localization error and the distances between the real and generated robot location. The mean, minimum, maximum and standard errors are calculated from absolute errors. Localization errors are the distances in meters calculated from location coordinates. Faster R-CNN system outperformed the AlexNet in terms of location and distance errors. Mean absolute errors of latitude and longitude from the Faster R-CNN method are slightly lower than the AlexNet, while minimum errors are also slightly lower in the case of Faster R-CNN. These small errors in latitude and longitude cause significant differences in distance errors. The average distance error of the Faster R-CNN is 28 m which is about half of the distance error of the AlexNet (~50 m). In the case of minimum errors, the distance error from Faster R-CNN is less than 1 m, while AlexNet has the minimum error around 3 m. On the maximum errors, the Faster R-CNN method has a distance error around 177 m, greatly less than the error from AlexNet of 322 m.

However, performances of Faster R-CNN localization suffer a decline in compass accuracy. On average, compass orientations from the Faster R-CNN method can have the errors around 55°, while the AlexNet gave an average error of 17°. The same trends went together with the minimum and maximum errors, in which the Faster R-CNN performed worse with larger compass errors.

The standard errors indicated that latitude and longitude coordinates resulted from all localization methods share the similar deviation, while the Faster R-CNN has higher compass differences, and the AlexNet have higher distance error differences in the results.

Table 5. Localization errors of proposed methods.

Errors		**Faster R-CNN**	**CNN (AlexNet)**
Mean Errors	Latitude	2.4367×10^{-4}	3.4441×10^{-4}
	Longitude	4.0868×10^{-5}	2.2187×10^{-4}
	Compass	54.9425	17.0498
	Distance (m)	28.4739	49.8166
Min Errors	Latitude	1.0000×10^{-6}	2.3391×10^{-6}
	Longitude	1.8654×10^{-7}	2.4863×10^{-7}
	Compass	0.3826	0.0374
	Distance (m)	0.5396	3.3838
Max Errors	Latitude	0.0011	0.0026
	Longitude	1.6409×10^{-4}	0.0013
	Compass	179.0098	173.2717
	Distance (m)	176.9496	321.9153
Standard Errors	Latitude	4.0797×10^{-5}	4.0797×10^{-5}
	Longitude	2.1783×10^{-6}	2.1783×10^{-6}
	Compass	6.0188	4.9259
	Distance (m)	1.4299	1.5464

4.3 Vision-Based Navigation System using Deep Q Network.

The navigation system was tested with simple navigation tasks in two environments: the simulation and the real outdoor environment. The robot had to move to the goal, which was randomly placed in front of the robot. The robot and the goal setups were the same as the training environment in Figure 3.13. There was a target area for random goal locations. The goal object was randomly placed in the target area. For the simulated experiments, the green armor in Fig. 3.17 (b) was used as the goal. The traffic cone in Fig. 3.17 (a) was used as the goal for the real robot. The starting position of the robot was the same as in the training environment. Simulation experiments included the navigation using the trained DQN. Real robot tests had an addition of the AR module to augment camera images. We measured the robot performance based on the reaching the target success rate and crash rate, since the rewards in the real environment cannot be measured in the same way as in the simulation. Success rate was measured by the times that our robot reached the goal. Crash rate was measured by the times that our robot moved into the grass Experimental results are presented in Table 6.

Table 6. Experimental results of the vision-based navigation system.

Measurements	**Experimental results**	
	Simulation	**Real robot**
Number of experiments	100	50
Success count	98	13
Success rate	98%	26%
Crash count	1	3
Crash rate	1%	6%
Maximum distance	10 m	4 m

4.3.1 Navigation in the Simulation Environment

The outdoor navigation system was tested inside the same simulation with the same simulation settings as in the training. We set the goal randomly in the target area. Goal distances were in the range of 5-10 m, same as in the training environment. The robot initial location is the at the same place, with the same orientation in every test. Goal locations were changed every time the test started. The robot had to move to the goal which was placed in front of it. Figure 4.3 shows an example of the navigation system in

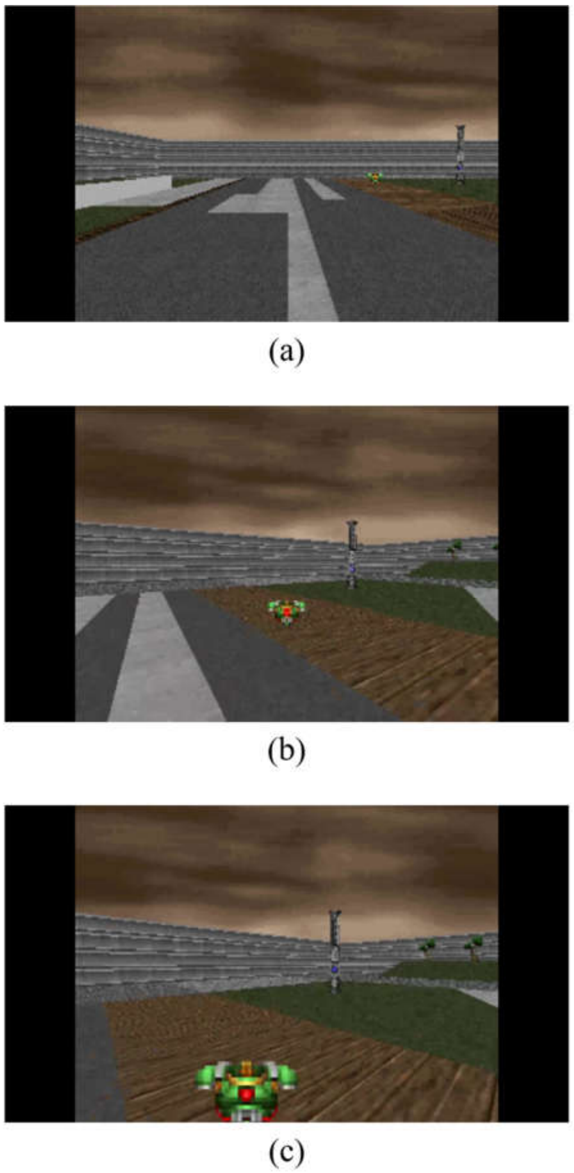

Figure 4.3: The agent reaching the goal object in the simulation at (a) starting; (b) midway; (c) goal reached.

the simulation, from the start to the goal. Our robot was tested in the simulation for 100 times. From the simulation results, the robot successfully reached the goal 98 times out of 100, or 98% success rate was achieved. The robot moved into the grass 1 time, which resulted in 1% crash rate. Another 1% was due to the time out, in which the robot moved over the designated step limit of 250 steps.

Though 98% success rate was achieved, there was a crash rate of 1% in simulation tests. It is suspected that the robot crashed because the robot lacks the knowledge of prior image frames. As implemented in [87], the DQN applied a 4-frame stack as an input for playing video games. However, we implemented only a single RGB image frame as the DQN input. The reason for the other fail to reach the goal due to the time out is related with the nature of DQN. Because DQN has a greedy policy as its core, it tried its best to achieve the most reward possible by avoiding taking risky actions that may cause large negative rewards.

4.3.2 Navigation in the Real Environment

Figure 4.4 shows the real experimental area, with the target area and the starting point included. We randomly placed the goal object in the target area. The robot started at the same location, with the same orientation in every test. Target area in real experiments was closer to the robot, in which the distances were reduced to 2-7 m instead of 5-10 m in simulation tests. Width of the target area was also changed to 7 m to match with the field of view in the robot camera. The closer distances were chosen to test the DQN from the shortest distance, which are typically easier for the robot. We gradually increased goal distances as the experiments proceed. The robot was tested for 50 times in the real

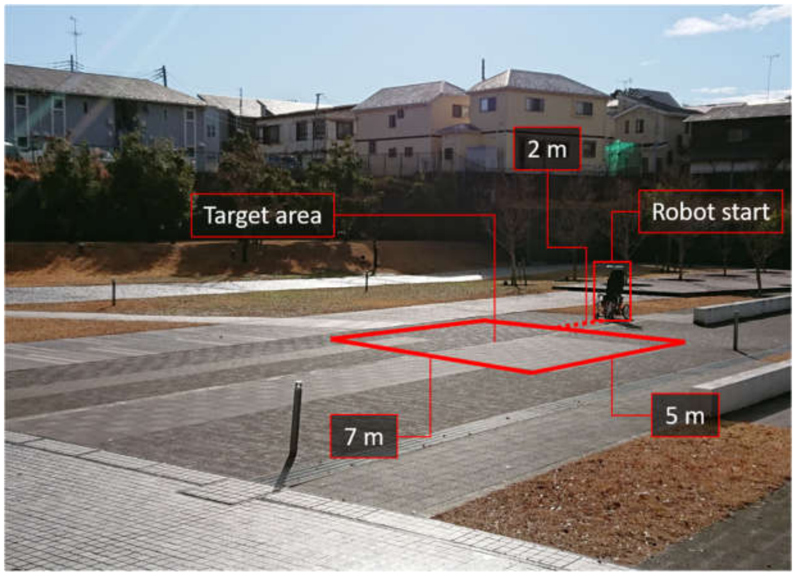

Figure 4.4: Experimental area in the real robot experiments.

environment. Augmented images from the camera were used as states. Robot performed an action according to the state and its knowledge from the simulation. A single action of the robot performed for 0.5 second before moving on to the next state and commit the next action.

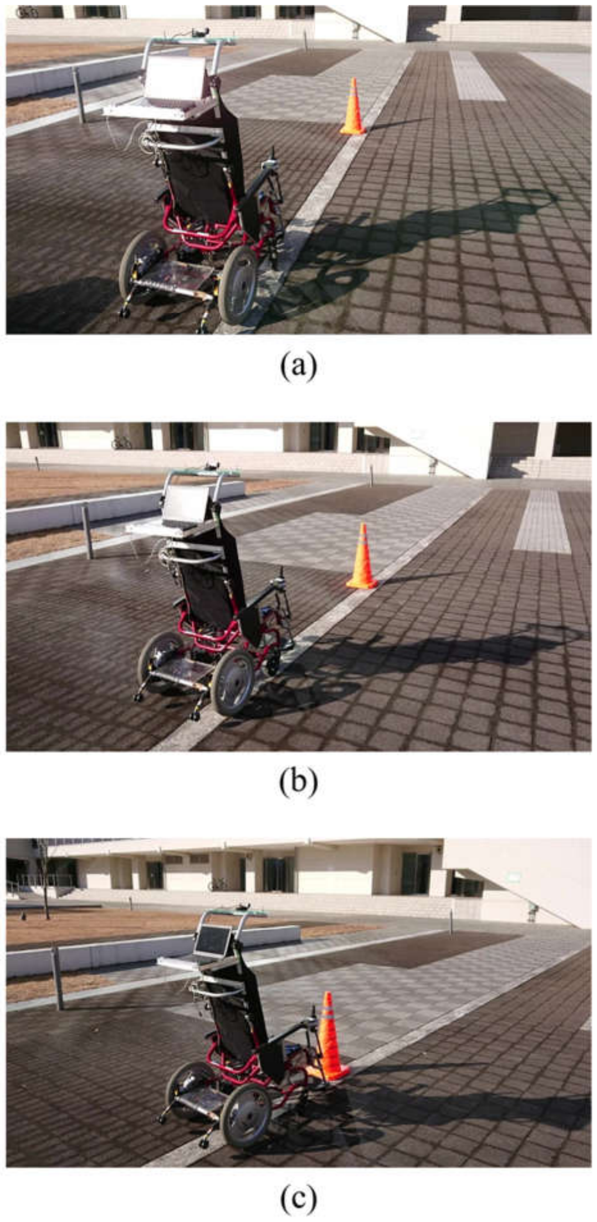

Figure 4.5: Wheelchair robot reaching the goal object at (a) 3 m, (b) 2 m and (c) reached the goal object.

From experimental results (Table 6), the robot successfully reached the goal object 13 times, resulted in 26% success rate. We counted the success when the robot reached the goal object, as displayed in Figure 4.5. The robot moved into the grass 3 times, which resulted in 6% crash rate. Other 68% of the real robot results were time out. The robot was designated with the time out status if the robot stuck in the same area for longer than 1 minute. All success attempts in the 26% success rate were from the navigation within the maximum distance of 4 m.

As seen in Figure 3.18, the robot could detect the goal object up to 10 m. However, the DQN could not guide the robot to the goal beyond 4 m. There are several reasons behind these results. First is the difference between the real environment and the simulation. Our DQN use convolutional neural networks to generate actions according to states, which is the whole augmented images that also include the background environments. This difference in backgrounds could reduce performances of the DQN. Secondly, the augmented reality module could possess several problems, since we employed a simple color-based object detector. Pedestrians could be mistaken as the goal object if they were wearing clothes with similar color to the object while walking into camera frames. Another problem could come from the DQN training. The training of DQN aims to have DQN generate and select actions based on action values. Some unseen states could cripple DQN performances by a large margin.

Despite the failure of 74%, the success rate of 26% showed that the simulation made from a video game can be used to train the DQN to guide the real robot for simple navigation tasks. However, assists from some other algorithms such as augmented reality, are highly recommended for real robot implementations.

4.4 Results from the Transfer Learning Implementation.

The task for the DQN for robot navigation is changed. The new task is introduced to the navigation system. Most of the environments were the same as the time of training the DQN for the simple navigation task. In this new task, the starting location and the goal location are random within the same designated zone which illustrated in Figure 4.6.

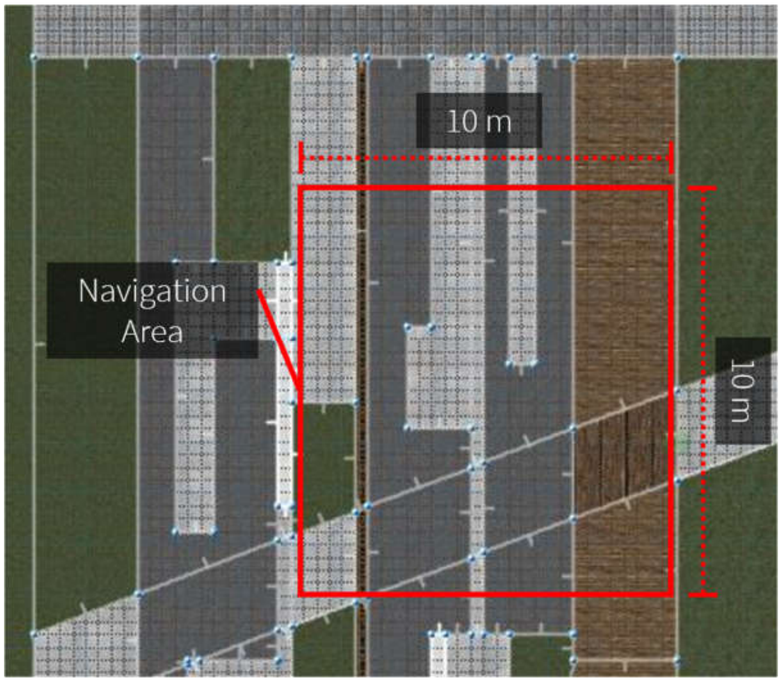

Figure 4.6: The new navigation zone.

The new zone is a square with 10 m of length per side. There are new obstacles included within the navigation. Obstacles are in the form of barrels of Figure 4.7. The task for the navigation system is to guide the robot from its starting location to the goal, which is the same green armor from the previous task. During its navigation, the robot must be able to avoid the barrels which are considered as obstacles. The locations of all obstacles are random within the navigation zone.

The transfer learning was implemented to train the new DQN agents by loading all the DQN parameters trained for the simple navigation task. The initial exploration factor was changed, to make the new DQN decide their actions during the training based on prior knowledge. Performances of our transfer learning strategy were tested with the new navigation task in the simulation environment.

We trained a reference DQN for the new navigation task without the use of any transfer learning strategies. This DQN was used as a performance reference point for other networks with transfer learning. The transfer learning strategy was employed in the training of other three DQN agents of the new task. These three new DQN agents contain

Figure 4.7: The obstacle (barrel).

the same training settings as the reference model, which are listed in Table 7. There are two differences in the settings. First, the number of training episodes in the DQN agents with transfer learning are reduced by half. Second, the initial values of the exploration factor are varied among three new models.

Table 7. Training settings for agents with transfer learning applications (A).

Training properties	**Reference**	**A1**	**A2**	**A3**
Episodes	1501	751		
Steps per episode	250			
Initial exploration parameter	1.0	0.72	0.60	0.48
Exploration decay	0.000048			
Learning rate	0.00024			
Reward discount	0.932			
Memory batch per training	96			
Memory size	1,000,000			

Four DQN agents were tested in the simulation environment for their designated task of reaching the goal with obstacle avoidance for 100 times. Results from the navigation task are listed in Table 8. These results include the success rate which counted from the success attempted in reaching the goal, the crash rate that counted each time the robot crashed into obstacles, and average scores that each agent received throughout 100 times of navigation tests.

Table 8. Results of the vision-based navigation systems with transfer learning.

Measurements	**Reference**	**Transfer Learning**		
		A1 (0.72)	**A2 (0.60)**	**A3 (0.48)**
Number of experiments	100	100	100	100
Success count	1	50	58	52
Success rate	1%	50%	58%	52%
Crash count	14	25	13	29
Crash rate	14%	25%	13%	29%
Average Score	- 1246.34	- 255.40	+ 104.88	- 215.38

The results stated that our transfer learning strategy helped in the training process of DQN agents. As the new navigation task is much more difficult than the simple reaching the goal, the reference model succeed only 1%. Agents with helps from the transfer learning performed far better than the reference, with better performances in all measurements. Among the new three agents, the DQN agent with balanced exploration and exploitation performed the best. The agent with 60% initial exploration factor balanced the use of prior knowledge, while also exploring new experiences.

It can be concluded from the results that transfer learning of the whole DQN agent helps the learning process of other agents with related tasks. Though the use of prior knowledge needs to be balanced between exploration and exploitation.

CHAPTER 5

CONCLUSIONS

This thesis is progressed to the final part. Matters in this thesis are concluded. We also suggest possible future works in this chapter.

5.1 Conclusion

In this thesis, we proposed three vision-based systems using deep learning algorithms for the localization and navigation tasks. The first system is the localization system based on landmarks in images. The second system is the navigation system using the Deep Q Network. The third system is the improvement of Deep Q Network, with implementations of a transfer learning strategy.

In the first localization system, Faster R-CNN was employed to detect landmarks in images. The detected landmarks and their properties which include the bounding boxes, labels, and detection scores were sent to the FFNN to generate the corresponding geolocation data of the image. The Faster R-CNN can detect landmarks in the real-time applications. Localization from the combination of the Faster R-CNN with the simple FFNN was proved to be more accurate than using a single state-of-the-art CNN model to localize the outdoor robots from images. In the best-case scenario, Faster R-CNN and FFNN can localize the robots in outdoor environments better than the GPS devices.

For the second system, Deep Q Network was employed for the vision-based navigation of an outdoor mobile robot. The DQN was trained in the game-based simulation environment to avoid possible damages from the training process. The implementation of the DQN on the real robot did not apply the traditional transfer learning. Instead, a marker-based AR algorithm was used to augment raw visual to fit with the trained experiences. A simple color detection algorithm was employed in the AR module. The trained DQN agent was able to guide the robot in the simulation flawlessly, while the navigation in the real robot was reduced in the navigation distances. The second system proved that game resources can be used to create practical simulation environments for outdoor mobile robots. Additionally, the AR algorithm can reduce unnecessary training procedures during the transition between the simulation and the real world.

The third system applied a transfer learning strategy to improve the training process of DQN agents. DQN is known for its training process with random actions through trial-and-error process. This training process can be drastically improved using knowledge of the existing DQN agents which do related tasks. The DQN for navigation trained with transfer learning strategy greatly improved in their performances. Their training can be reduced shorter, which save a large amount of computation time and resources.

All the system proposed showed performances of supervised and unsupervised deep learning algorithms for the practical localization and navigation tasks of outdoor mobile robots. Regardless of the category, deep learning algorithms can be practically employed for robotics tasks. Performances of deep learning algorithms are related to how they are prepared, and the tasks they are up to.

5.2 Future works

From three systems proposed for the outdoor mobile robots, there are many issues left for the future works. This section separates the possibilities by the system.

1) **Faster R-CNN for landmark-based localization:** Even though good performances were inspected from the system, there are plenty of properties to be improved. There are newer object detectors available. Some of those detector yield benefits of speed and accuracy. The transition of knowledge between types of landmarks in different areas is also an interesting topic.

2) **Deep Q Network for Vision-Based Outdoor Navigation:** DQN is a greedy algorithm. It does not possess any kind of policy within. It is wise to try and

implement on-policy algorithms such as A3C. The detection system for the AR module should also be changed to the algorithms based on features such as SIFT.

3) **Transfer Learning for DQN Improvements:** Transfer learning is one of the most interesting topics for the field of unsupervised learning. There are many different ways to transfer knowledge from one model to another. Since the whole model is transferred in this thesis, it is nice to see the selective transferring algorithm for deep reinforcement learning algorithms. Some algorithms such as Genetic Algorithms (GA) can be used to select parts of the model to transfer, or which features to transfer.

REFERENCES

[1] S. Roland, R.N Illah, “Mobile Robot Localization,” in *Introduction to Autonomous Mobile Robots*, 1st ed., Bradford Company Scituate: Cambridge, MA, USA, 2004, pp. 181–256.

[2] Jihong Lee et al., "Operating a six-legged outdoor patrol robot," *2007 International Conference on Control, Automation and Systems*, Seoul, 2007, pp. 1034-1039.

[3] Z. Hao and S. Huang, "Integrated Navigation System Based on Differential Magnetic Compass and GPS," *2009 International Conference on Information Engineering and Computer Science*, Wuhan, 2009, pp. 1-4.

[4] D. Pazderski and P. Dutkiewicz, "Low-cost GPS receivers in navigation of mobile robots," *Proceedings of the Third International Workshop on Robot Motion and Control, 2002. RoMoCo '02.*, Bukowy Dworek, Poland, 2002, pp. 119-122.

[5] F. van Diggelen and P. Enge, “The World’s first GPS MOOC and Worldwide Laboratory using Smartphones,” *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015)*, Tampa, Florida, September 2015, pp. 361-369.

[6] M. B. Kjærgaard et al, “Indoor Positioning Using GPS Revisited,” *Pervasive Computing*, vol. 6030, Heidelberg: Springer, 2010, pp. 38–56.

[7] J. Velagic, N. Osmic, F. Hodzic and H. Siljak, "Outdoor navigation of a mobile robot using GPS and GPRS communication system," *Proceedings ELMAR-2011*, Zadar, 2011, pp. 173-177.

[8] K. M. Al-Aubidy, M. M. Ali, A. M. Derbas and A. W. Al-Mutairi, "GPRS-based remote sensing and teleoperation of a mobile robot," *10th International Multi-Conferences on Systems, Signals & Devices 2013 (SSD13)*, Hammamet, 2013, pp. 1-7.

[9] L. Guenda, L. Brás, M. Oliveira and N. B. Carvalho, "Indoor/outdoor management system compliant with Google Maps and Android® OS," *2011 IEEE EUROCON - International Conference on Computer as a Tool*, Lisbon, 2011, pp. 1-4.

[10] D. Tian, X. He, L. Zhang, J. Lian and X. Hu, "A Design of Odometer-Aided Visual Inertial Integrated Navigation Algorithm Based on Multiple View Geometry Constraints," *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Hangzhou, 2017, pp. 161-166.

[11] C. Chen, W. Chai, A. K. Nasir and H. Roth, "Low cost IMU based indoor mobile robot navigation with the assist of odometry and Wi-Fi using dynamic constraints," *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, Myrtle Beach, SC, 2012, pp. 1274-1279.

[12] I. Ohya, A. Kosaka and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing," in *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 969-978, Dec. 1998.

[13] Y. Fan, G. Cui and F. Lei, "Application of Edge Detection Algorithm Based on Morphology in Robot Vision System," *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, Zhejiang, 2009, pp. 304-307.

[14] Lixin Tang and S. Yuta, "Vision based navigation for mobile robots in indoor environment by teaching and playing-back scheme," *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, Seoul, South Korea, 2001, pp. 3072-3077 vol.3.

[15] W. Shi and J. Samarabandu, "Corridor Line Detection for Vision Based Indoor Robot Navigation," *2006 Canadian Conference on Electrical and Computer Engineering*, Ottawa, Ont., 2006, pp. 1988-1991.

[16] Z. Zhang, Y. Jiang, C. Zhang, C. Zhang and X. Li, "The Optimization of Localization and Navigation for Vision-based Robot," *2019 IEEE International*

Conference on Integrated Circuits, Technologies and Applications (ICTA), Chengdu, China, 2019, pp. 180-181.

[17] C. H. Yun, Y. Moon and N. Y. Ko, "Vision based navigation for golf ball collecting mobile robot," *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, Gwangju, 2013, pp. 201-203.

[18] W. Chang and P. Chu, "An intelligent space for mobile robot navigation with on-line calibrated vision sensors," *2010 11th International Conference on Control Automation Robotics & Vision*, Singapore, 2010, pp. 1452-1457.

[19] A. Bur, A. Tapus, N. Ouerhani, R. Siegwart and H. Hugli, "Robot Navigation by Panoramic Vision and Attention Guided Fetaures," *18th International Conference on Pattern Recognition (ICPR'06)*, Hong Kong, 2006, pp. 695-698.

[20] M. Sharifi and XiaoQi Chen, "A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards," *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, Queenstown, 2015, pp. 251-255.

[21] C. Jiaqian, H. Yan and J. Jingping, "Calibration-free Visual Navigation of Robot in a Maze," *2006 IEEE International Conference on Information Acquisition*, Weihai, 2006, pp. 65-69.

[22] Tatsuhiko Hirukawa, Satoshi Komada and Junji Hirai, "Image Feature based Navigation of Nonholonomic Mobile Robots with Active Camera," *SICE Annual Conference 2007*, Takamatsu, 2007, pp. 2502-2506.

[23] K. Bhongale and S. Gore, "Design of robot navigation monitoring system using image feature analysis and omnidirectional camera images," *2017 2nd International Conference for Convergence in Technology (I2CT)*, Mumbai, 2017, pp. 405-409.

[24] W. Wang et al., "A ceiling feature-based vision control system for a service robot," *2017 36th Chinese Control Conference (CCC)*, Dalian, 2017, pp. 6614-6619.

[25] Y. Zhai, G. Wang and J. Xu, "Real-time Obstacle Map Reconstruction by Sensors with Limited Field of View for Vision Based Autonomous Robot," *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, Xiamen, China, 2018, pp. 1-5.

[26] K. Sharma, K. Jeong and S. Kim, "Vision based autonomous vehicle navigation with self-organizing map feature matching technique," *2011 11th International Conference on Control, Automation and Systems*, Gyeonggi-do, 2011, pp. 946-949.

[27] M. S. Guzel and P. Nattharith, "New Technique for distance estimation using SIFT for mobile robots," *2014 International Electrical Engineering Congress (iEECON)*, Chonburi, 2014, pp. 1-4.

[28] J. Liu, M. Nie, H. Wu and X. Mai, "An image-based Accurate Alignment for Substation Inspection Robot," *2018 International Conference on Power System Technology (POWERCON)*, Guangzhou, 2018, pp. 4113-4117.

[29] S. Yue-hua and c. Yuan-Ii, "Image Feature Extraction for Vision-Based UAV Navigation," *2018 Chinese Automation Congress (CAC)*, Xi'an, China, 2018, pp. 1130-1134.

[30] Pifu Zhang, E. E. Milios and J. Gu, "Vision data registration for robot self-localization in 3D," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alta., 2005, pp. 2315-2320.

[31] Y. Kim, J. Lee, M. S. Aldosari, M. S. Altokhais and J. Lee, "Vision-based corridor path search of a mobile robot," *2011 11th International Conference on Control, Automation and Systems*, Gyeonggi-do, 2011, pp. 700-705.

[32] Seung-Youn Lee and D. Kwak, "A terrain classification method for UGV autonomous navigation based on SURF," *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Incheon, 2011, pp. 303-306.

[33] L. Guohua, X. Xiandong, Y. Xiang, W. Yadong and Q. Tianwei, "An Indoor Localization Method for Humanoid Robot Based on Artificial Landmark," *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, Qinhuangdao, 2015, pp. 1854-1857.

[34] Y. Hagiwara, T. Inamura and Y. Choi, "Effectiveness evaluation of view-based navigation for obstacle avoidance," *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, Gwangju, 2013, pp. 1029-1033.

[35] R. O. Prakash and C. Saravanan, "Autonomous robust helipad detection algorithm using computer vision," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, 2016, pp. 2599-2604.

[36] P. Wu, H. Zhang and R. Du, "Research on binocular vision-aided inertial navigation system," *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, Beijing, 2014, pp. 1-6.

[37] T. Wang, H. Huang, J. Lin, C. Hu, K. Zeng and M. Sun, "Omnidirectional CNN for Visual Place Recognition and Navigation," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, 2018, pp. 2341-2348.

[38] C. Park, J. Jang, L. Zhang and J. Jung, "Light-weight visual place recognition using convolutional neural network for mobile robots," *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, 2018, pp. 1-4.

[39] Z. Li, A. Zhou, M. Wang and Y. Shen, "Deep Fusion of Multi-Layers Salient CNN Features and Similarity Network for Robust Visual Place Recognition*," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, China, 2019, pp. 22-29.

[40] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature 521*, 436–444 (2015).

[41] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[42] A. Khasnobish et al., "Object-shape recognition from tactile images using a feed-forward neural network," *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Brisbane, QLD, 2012, pp. 1-8.

[43] Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), pp.504-507.

[44] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. Association for Computing Machinery, New York, NY, USA, 609–616.

[45] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[46] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[47] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[48] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 580-587.

[49] R. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 1440-1448.

[50] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 1 June 2017.

[51] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 779-788.

[52] U. Jänen, C. Paul, M. Wittke and J. Hähner, "Multi-object tracking using feed-forward neural networks," *2010 International Conference of Soft Computing and Pattern Recognition*, Paris, 2010, pp. 176-181.

[53] J. G. Rázuri, D. Sundgren, R. Rahmani and A. M. Cardenas, "Automatic Emotion Recognition through Facial Expression Analysis in Merged Images Based on an Artificial Neural Network," *2013 12th Mexican International Conference on Artificial Intelligence*, Mexico City, 2013, pp. 85-96.

[54] X. Chen, S. Xiang, C. Liu and C. Pan, "Aircraft Detection by Deep Belief Nets," *2013 2nd IAPR Asian Conference on Pattern Recognition*, Naha, 2013, pp. 54-58.

[55] W. Diao, X. Sun, X. Zheng, F. Dou, H. Wang and K. Fu, "Efficient Saliency-Based Object Detection in Remote Sensing Images Using Deep Belief Networks," in *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 2, pp. 137-141, Feb. 2016.

[56] D. Liang, K. Weng, C. Wang, G. Liang, H. Chen and X. Wu, "A 3D object recognition and pose estimation system using deep learning method," *2014 4th IEEE International Conference on Information Science and Technology*, Shenzhen, 2014, pp. 401-404.

[57] L. Zhao, Z. Wang, X. Wang and Q. Liu, "Driver drowsiness detection using facial dynamic fusion information and a DBN," in *IET Intelligent Transport Systems*, vol. 12, no. 2, pp. 127-133, 3 2018.

[58] S. Kamada and T. Ichimura, "An Object Detection by using Adaptive Structural Learning of Deep Belief Network," *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019, pp. 1-8.

[59] Ying-Nong Chen, Chin-Chuan Han, Cheng-Tzu Wang, Bor-Shenn Jeng and Kuo-Chin Fan, "The Application of a Convolution Neural Network on Face and License Plate Detection," *18th International Conference on Pattern Recognition (ICPR'06)*, Hong Kong, 2006, pp. 552-555.

[60] M. Szarvas, U. Sakai and Jun Ogata, "Real-time Pedestrian Detection Using LIDAR and Convolutional Neural Networks," *2006 IEEE Intelligent Vehicles Symposium*, Tokyo, 2006, pp. 213-218.

[61] H. Zong, L. Bao, B. Liu and J. Qiu, "Application of Convolutional Neural Network in Target Detection of Millimeter Wave Imaging," *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Boston, MA, 2018, pp. 1217-1218.

[62] S. Mane and S. Mangale, "Moving Object Detection and Tracking Using Convolutional Neural Networks," *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2018, pp. 1809-1813.

[63] M. F. Haque, H. Lim and D. Kang, "Object Detection Based on VGG with ResNet Network," *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-3.

[64] Y. Li, J. Zhao, S. Zhang and W. Tan, "Aircraft Detection in Remote Sensing Images Based on Deep Convolutional Neural Network," *2018 IEEE 3rd International Conference on Cloud Computing and Internet of Things (CCIOT)*, Dalian, China, 2018, pp. 135-138.

[65] K. Saleh, M. Hossny, A. Hossny and S. Nahavandi, "Cyclist detection in LIDAR scans using faster R-CNN and synthetic depth images," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, 2017, pp. 1-6.

[66] H. Zhang, Y. Du, S. Ning, Y. Zhang, S. Yang and C. Du, "Pedestrian Detection Method Based on Faster R-CNN," *2017 13th International Conference on Computational Intelligence and Security (CIS)*, Hong Kong, 2017, pp. 427-430.

[67] R. Wang, Y. You, Y. Zhang, W. Zhou and J. Liu, "Ship detection in foggy remote sensing image via scene classification R-CNN," *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Guiyang, 2018, pp. 81-85.

[68] H. Guan, Y. Yu, J. Li and P. Liu, "Pole-Like Road Object Detection in Mobile LiDAR Data via Supervoxel and Bag-of-Contextual-Visual-Words Representation," in *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 4, pp. 520-524, April 2016.

[69] C. Zhihong, Z. Hebin, W. Yanbo, L. Binyan and L. Yu, "A vision-based robotic grasping system using deep learning for garbage sorting," *2017 36th Chinese Control Conference (CCC)*, Dalian, 2017, pp. 11223-11226.

[70] F. H. Zunjani, S. Sen, H. Shekhar, A. Powale, D. Godnaik and G. C. Nandi, "Intent-based Object Grasping by a Robot using Deep Learning," *2018 IEEE 8th International Advance Computing Conference (IACC)*, Greater Noida, India, 2018, pp. 246-251.

[71] Y. Yoshimoto and H. Tamukoh, "Object Recognition System using Deep Learning with Depth Images for Service Robots," *2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Ishigaki, Okinawa, Japan, 2018, pp. 436-441.

[72] J. Akbar, M. Shahzad, M. I. Malik, A. Ul-Hasan and F. Shafait, "Runway Detection and Localization in Aerial Images using Deep Learning," *2019 Digital*

Image Computing: Techniques and Applications (DICTA), Perth, Australia, 2019, pp. 1-8.

[73] H. Chen, W. Chiu, J. Yu, H. Chen and W. Wang, "The obstacles detection for outdoor robot based on computer vision in deep learning," *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, Berlin, Germany, 2019, pp. 184-188.

[74] K. M. Ibrahim Khalilullah, S. Ota, T. Yasuda and M. Jindai, "Development of robot navigation method based on single camera vision using deep learning," *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, 2017, pp. 939-942.

[75] T. D. Le, D. T. Huynh and H. V. Pham, "Efficient Human-Robot Interaction using Deep Learning with Mask R-CNN: Detection, Recognition, Tracking and Segmentation," *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Singapore, 2018, pp. 162-167.

[76] G. V. Konoplich, E. O. Putin and A. A. Filchenkov, "Application of deep learning to the problem of vehicle detection in UAV images," *2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM)*, St. Petersburg, 2016, pp. 4-6.

[77] W. Budiharto, A. A. S. Gunawan, J. S. Suroso, A. Chowanda, A. Patrik and G. Utama, "Fast Object Detection for Quadcopter Drone Using Deep Learning," *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, Nagoya, 2018, pp. 192-195.

[78] Y. Chao, X. Chen and N. Xiao, "Deep learning-based grasp-detection method for a five-fingered industrial robot hand," in *IET Computer Vision*, vol. 13, no. 1, pp. 61-70, 2 2019.

[79] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio and G. Sansoni, "Deep Learning Based Machine Vision: First Steps Towards a Hand Gesture Recognition Set Up

for Collaborative Robots," *2018 Workshop on Metrology for Industry 4.0 and IoT*, Brescia, 2018, pp. 28-33.

[80] P. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano and T. Ogata, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," *in IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397-403, April 2017.

[81] Hang Su, Yusi Zhang, Jingsong Li and Jie Hu, "The shopping assistant Robot design based on ROS and deep learning," *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT)*, Dalian, 2016, pp. 173-176.

[82] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285.

[83] van Otterlo, M., Wiering, M., “Reinforcement learning and markov decision processes,” *Reinforcement Learning. Adaptation, Learning, and Optimization*, vol. 12, pp. 3–42, 2012.

[84] Tokic M., Palm G. (2011) Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax. In: *Bach J., Edelkamp S. (eds) KI 2011: Advances in Artificial Intelligence. KI 2011. Lecture Notes in Computer Science, vol 7006*. Springer, Berlin, Heidelberg.

[85] S. Fujimoto, D. Meger, D. Precup, “Off-Policy Deep Reinforcement Learning without Exploration,” arXiv:1812.02900.

[86] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press.

[87] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, 2015, pp. 529–533.

[88] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the*

33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16). JMLR.org, pp. 1928–1937.

[89] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 2094–2100.

[90] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1995–2003.

[91] R. Chen and X. Dai, "Robotic Grasp Control Policy with Target Pre-detection Based on Deep Q-learning," *2018 3rd International Conference on Robotics and Automation Engineering (ICRAE)*, Guangzhou, 2018, pp. 29-33.

[92] E. Bejar and A. Morán, "Backing Up Control of a Self-Driving Truck-Trailer Vehicle with Deep Reinforcement Learning and Fuzzy Logic," *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Louisville, KY, USA, 2018, pp. 202-207.

[93] R. Özaln, C. Kaymak, Ö. Yildirum, A. Ucar, Y. Demir and C. Güzeliş, "An Implementation of Vision Based Deep Reinforcement Learning for Humanoid Robot Locomotion," *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, Sofia, Bulgaria, 2019, pp. 1-5.

[94] J. Kim, B. Kim, J. Yoon, M. Lee, S. Jung and J. y. Choi, "Robot Soccer Using Deep Q Network," *2018 International Conference on Platform Technology and Service (PlatCon)*, Jeju, 2018, pp. 1-6.

[95] H. Kim, D. Seo and D. Kim, "Push Recovery Control for Humanoid Robot Using Reinforcement Learning," *2019 Third IEEE International Conference on Robotic Computing (IRC)*, Naples, Italy, 2019, pp. 488-492.

[96] X. Xue, Z. Li, D. Zhang and Y. Yan, "A Deep Reinforcement Learning Method for Mobile Robot Collision Avoidance based on Double DQN," *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, Vancouver, BC, Canada, 2019, pp. 2131-2136.

[97] H. Bui and N. Y. Chong, "Autonomous Speech Volume Control for Social Robots in a Noisy Environment Using Deep Reinforcement Learning*," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, China, 2019, pp. 1263-1268.

[98] K. Lobos-Tsunekawa, F. Leiva and J. Ruiz-del-Solar, "Visual Navigation for Biped Humanoid Robots Using Deep Reinforcement Learning," in *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247-3254, Oct. 2018.

[99] W. Li, D. Chen and J. Le, "Robot Patrol Path Planning Based on Combined Deep Reinforcement Learning," *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, Melbourne, Australia, 2018, pp. 659-666.

[100] P. Ciou, Y. Hsiao, Z. Wu, S. Tseng and L. Fu, "Composite Reinforcement Learning for Social Robot Navigation," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, 2018, pp. 2553-2558.

[101] Q. Zhang, J. Lin, Q. Sha, B. He and G. Li, "Deep Interactive Reinforcement Learning for Path Following of Autonomous Underwater Vehicle," in *IEEE Access*, vol. 8, pp. 24258-24268, 2020.

[102] X. Qiu, K. Wan and F. Li, "Autonomous Robot Navigation in Dynamic Environment Using Deep Reinforcement Learning," *2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, Shenyang, China, 2019, pp. 338-342.

[103] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, Shanghai, 2017, pp. 68-72.

[104] X. Ruan, D. Ren, X. Zhu and J. Huang, "Mobile Robot Navigation based on Deep Reinforcement Learning," *2019 Chinese Control And Decision Conference (CCDC)*, Nanchang, China, 2019, pp. 6174-6178.

[105] P. Yue, J. Xin, H. Zhao, D. Liu, M. Shan and J. Zhang, "Experimental Research on Deep Reinforcement Learning in Autonomous navigation of Mobile Robot," *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA*), Xi'an, China, 2019, pp. 1612-1616.

[106] S. Han, H. Choi, P. Benz and J. Loaiciga, "Sensor-Based Mobile Robot Navigation via Deep Reinforcement Learning," *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Shanghai, 2018, pp. 147-154.

[107] N. Duo, Q. Wang, Q. Lv, H. Wei and P. Zhang, "A Deep Reinforcement Learning Based Mapless Navigation Algorithm Using Continuous Actions," *2019 International Conference on Robots & Intelligent System (ICRIS)*, Haikou, China, 2019, pp. 63-68.

[108] C. Wang, J. Wang, X. Zhang and X. Zhang, "Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning," *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Montreal, QC, 2017, pp. 858-862.

[109] J. Kulhánek, E. Derner, T. de Bruin and R. Babuška, "Vision-based Navigation Using Deep Reinforcement Learning," *2019 European Conference on Mobile Robots (ECMR)*, Prague, Czech Republic, 2019, pp. 1-8.

[110] Singh, S.P. Transfer of learning by composing solutions of elemental sequential tasks. *Mach Learn 8*, 323–339 (1992).

[111] Caruana, R. Multitask Learning. *Machine Learning 28*, 41–75 (1997).

[112] L. Shao, F. Zhu and X. Li, "Transfer Learning for Visual Categorization: A Survey," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019-1034, May 2015.

[113] S. Jiang, Y. Kong and Y. Fu, "Deep Geo-Constrained Auto-Encoder for Non-Landmark GPS Estimation," in *IEEE Transactions on Big Data*, vol. 5, no. 2, pp. 120-133, 1 June 2019.

[114] Z. Huang, C. He, Z. Wang, J. Xi, H. Wang and L. Hou, "Cinnamomum Camphora Classification Based on Leaf Image Using Transfer Learning," *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chengdu, China, 2019, pp. 1426-1429.

[115] R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning," *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018, pp. 1-4.

[116] Z. Li, R. Togo, T. Ogawa and M. Haseyama, "Classification of Subcellular Protein Patterns in Human Cells with Transfer Learning," *2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech)*, Osaka, Japan, 2019, pp. 273-274.

[117] S. Pelletier, A. Montacir, H. Zakari and M. Akhloufi, "Deep Learning for Marine Resources Classification in Non-Structured Scenarios: Training vs. Transfer Learning," *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, Quebec City, QC, 2018, pp. 1-4.

[118] R. Dalal and T. Moh, "Fine-Grained Object Detection Using Transfer Learning and Data Augmentation," *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Barcelona, 2018, pp. 893-896.

[119] N. Huber-fliflet, F. Wei, H. Zhao, H. Qin, S. Ye and A. Tsang, "Image Analytics for Legal Document Review : A Transfer Learning Approach," *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, 2019, pp. 4325-4328.

[120] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 7961-7967.

[121] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, 2009.

[122] K. Shao, Y. Zhu and D. Zhao, "StarCraft Micromanagement With Reinforcement Learning and Curriculum Transfer Learning," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 73-84, Feb. 2019.

[123] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart and J. Nieto, "Comparing Task Simplifications to Learn Closed-Loop Object Picking Using Deep Reinforcement Learning," in *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1549-1556, April 2019.

[124] J. v. Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres and D. Nikovski, "Sim-to-Real Transfer Learning using Robustified Controllers in Robotic Tasks involving Complex Dynamics," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, pp. 6001-6007.

[125] W. Yu, V. C. Kumar, G. Turk and C. K. Liu, "Sim-to-Real Transfer for Biped Locomotion," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 3503-3510.

[126] X. B. Peng, M. Andrychowicz, W. Zaremba and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, 2018, pp. 3803-3810.

[127] J. Karttunen, A. Kanervisto, V. Kyrki and V. Hautamäki, "From Video Game to Real Robot: The Transfer Between Action Spaces," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 3567-3571.

[128] B. Qin, Y. Gao and Y. Bai, "Sim-to-real: Six-legged Robot Control with Deep Reinforcement Learning and Curriculum Learning," *2019 4th International Conference on Robotics and Automation Engineering (ICRAE)*, Singapore, Singapore, 2019, pp. 1-5.

[129] H. Bharadhwaj, Z. Wang, Y. Bengio and L. Paull, "A Data-Efficient Framework for Training and Sim-to-Real Transfer of Navigation Policies," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, pp. 782-788.

[130] F. Zhu, L. Zhu and Y. Yang, "Sim-Real Joint Reinforcement Transfer for 3D Indoor Navigation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 11380-11389.

[131] S. Nilwong, D. Hossain, S. I. Kaneko, G. Capi, "Deep Learning-Based Landmark Detection for Mobile Robot Outdoor Localization," *Machines*, vol. 7, issue 2, 2019, pp. 1-14.

[132] Bayar, B.; Stamm, M.C. Design Principles of Convolutional Neural Networks for Multimedia Forensics. *Electron. Imaging Med. Watermark. Secur. Forensics* 2017, 10, 77–86.

[133] S. Nilwong, G. Capi, " Outdoor Robot Navigation System using Game-Based DQN and Augmented Reality," *2020 17th International Conference on Ubiquitous Robots (UR2020)*, Kyoto, Japan, 2020.

[134] M. Kempka, M. Wydmuch, G. Runc, J. Toczek and W. Jaśkowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, Santorini, 2016, pp. 1-8.

[135] J. W. Anderson, Doom Builder: An Illustrated Guide, July 2004.

[136] I. Ullah and A. Petrosino, “About pyramid structure in convolutional neural networks,” *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, 2016, pp. 1318-1324.

[137] R. Yang, “The study and improvement of Augmented reality based on feature matching,” IEEE 2nd International Conference on Software Engineering and Service Science, Beijing, 2011, pp. 586-589.

[138] D. Hossain, G. Capi, M. Jindai and S. Kaneko, “Pick-place of dynamic objects by robot manipulator based on deep learning and easy user interface teaching systems,” Industrial Robot, vol. 44, issue 1, pp. 11-20, January 2017.

[139] Alex, K.; Ilya, S.; Geoffrey E.H. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS’12)*, Lake Tahoe, NV, USA, 3–6 December 2012.

[140] Mamajek, E.E.; Prsa, A.; Torres, G.; Harmanec, P.; Asplund, M.; Bennett, P.; Capitaine, N.; Christensen-Dalsgaard, J.; Depagne, É.; Folkner, M.W.; et al. Resolution B3 on Recommended Nominal Conversion Constants for Selected Solar and Planetary Properties. In *Proceedings of the 29th IAU General Assembly (IAU 2015)*, Honolulu, HI, USA, 3–14 August 2015.