Internship – Robotic Vision for the Pepper Robot

Final Reflective Report

1. Company Introduction

The host organization for this internship is IRL CROSSING, where IRL (short for international research laboratory) is a flagship international collaboration mechanism used by CNRS (The French National Centre for Scientific Research). The lab is relatively young (launched in early 2021) and directed by Professor Jean-Philippe Diguet. It has partnership with a number of research institutions including CNRS, University of Adelaide, University of South Australia, Flinders University, IMT Atlantique, and an industrial partner Naval Group, which makes it one of the world's only five IRLs with industrial partners.

IRL CROSSING focuses on the research area at the interface of computer science, artificial intelligence, technology, engineering, human factors and psychology. They have a range of research projects that lie in those areas such as developing solutions for light pollutions on console displays, real-time monitoring of human performance, predictive maintenance with human experts in the loop, detecting user's implicit need of help for human-robot interaction, adaptive control of underwater robots with machine learning, learning by robot manipulation demonstration via generative adversarial networks (GANs), perceiving distant collaborative activity using mixed reality, and human performance in maritime environment. They have recruited a number of PhD, master's, and internship students from the partner universities to work on those projects. The main goals of their research include: 1) to improve control systems (e.g., robots) to assist and enhance human performance, and 2) to improve the way human operators interact with these control systems.

As mentioned above, IRL CROSSING is an inter-disciplinary research lab. They value the knowledge of researchers from different disciplines and encourage inter-disciplinary collaborations. They also have a very inclusive culture where researchers and students of different cultural backgrounds and characteristics are all appreciated and welcomed.

2. Tasks I Carried Out

In line with the company's vision in improving human-AI interaction, they have a team of students participating in the competition "RoboCup@Home" held in July 2022 which is a competition aiming to develop assistive robot in domestic applications. The robot used in the competition is called "Pepper" which is manufactured by Softbank Robotics. The scenario of the competition is a party held in a house, where the robot needs to help the guests (e.g., bring drinks to guests). My overall task is to help implement the functionalities related to computer vision on the robot. The main sub-tasks I completed are described below.

2.1 Object detection

My first task was to implement an object detector on the robot. Since training of an object detector (including collecting/annotating data) would take considerable time, we decided to use pre-trained object

detection models. The model needs to be able to detect as many household objects (e.g., furniture, kitchenware) as possible with reasonable accuracy. I implemented the YOLOv3 object detection model that is pre-trained on the Microsoft COCO dataset and able to detect 80 object classes. Although the model has good performance on the PC, it is relatively large in size and would cause significant delay when implemented on the robot due to the robot's limited processing power. Therefore, we decided to use the lightweight MobileNet model instead which is able to detect around 600 object classes (Figure 1).

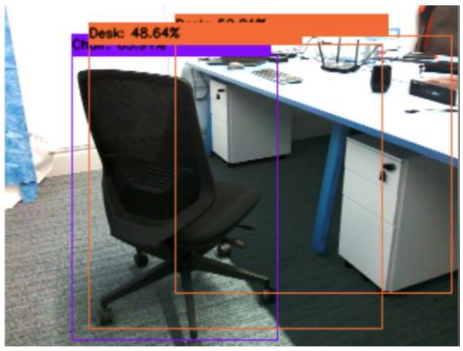

Figure 1. Example object detection by the MobileNet model. In this example, the model captured the chair with reasonable accuracy. However, the bounding box for the desk was not very accurate since it also captured the chair.

2.2 Color detection

After an object is detected, the robot needs to detect its main color in order to describe the object and distinguish between different objects of the same category. For example, if the robot is required to hand a red backpack to a guest, the robot needs to recognize the backpack in red color from several backpacks in different colors. In the competition, the color of an object is assumed to be mostly uniform. Based on this assumption, I implemented the color detection feature using the k-means clustering algorithm on the area inside the bounding box for a detected object. The data used for clustering can either be the RGB values or the Hue values of the pixels, which give similar results. Based on a number of experiments, k=5 gives correct color detection for most objects. The resulting color names were simplified (e.g., different shades of blue are all simplified to "blue") to be easily understood. Furthermore, due to the high dimensionality of the raw images and the limited processing power of the robot, the images were downsampled to 15% of the original resolution to achieve a reasonable speed. As shown in Figure 2, if the bounding box for an object is accurate, this method can correctly detect the color of the object.

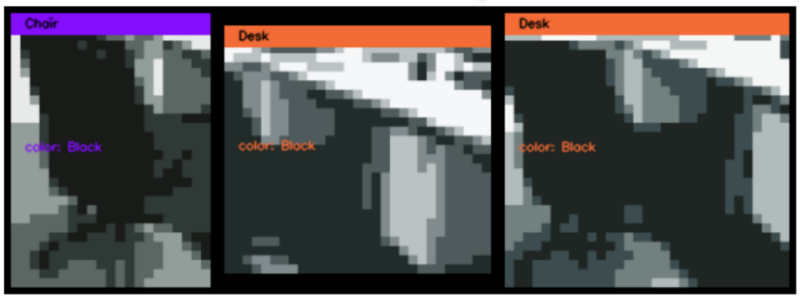

Figure 2. Example results for color detection. For each bounding box, the RGB pixel values are clustered into five clusters using k-means algorithm. The raw pixel values are then replaced by the RGB values of their corresponding cluster centers. The corresponding raw input image is shown in Figure 1. In this example, the color for the chair was predicted correctly, whereas the color prediction for the desk was wrong due to the inaccurate bounding boxes.

2.3 Visualization

2.3.1 Rviz

After the objects are detected, the position of the objects relative to the front camera (by which the RGB images are acquired) can be estimated using the pinhole model. It is important to make sure the positions of the objects are correctly estimated as the robot might be required to grasp certain objects. To confirm this, I used the rviz (short for ROS visualization) software to visualize the robot's perception of the world and its position relative to the surrounding objects. The pixel at the center of the bounding box was used to estimate the position of an object in the coordinate system of the robot's front camera frame. This position is then transformed to the coordinate system of the base footprint frame using transformation matrices (Figure 3).

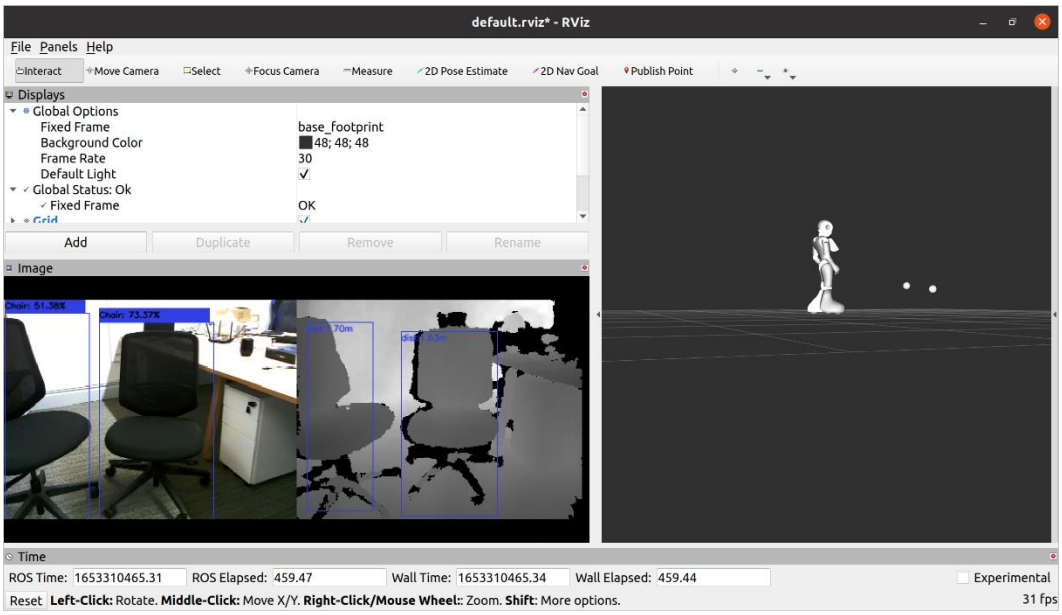

Figure 3. Example rviz visualization for object detection. In this example, two chairs were detected (left), and their relative position to the robot was visualized in 3D (right).

2.3.2 VizBox

VizBox is a web application developed specifically for the RoboCup@Home competition using the web framework "tornado". During the competition, the teams are required to show the robot's perception and its communication with the operator in real time to the audience. The source code of VizBox is public but contains minor bugs. The participants are allowed to make modifications to the source code while keeping the user interface original. I first fixed the bugs in the source code to make it work in Python 2. Furthermore, as our team was migrating from Python 2 to Python 3 and tornado is implemented differently in Python 3, I made an updated version of VizBox for Python 3.

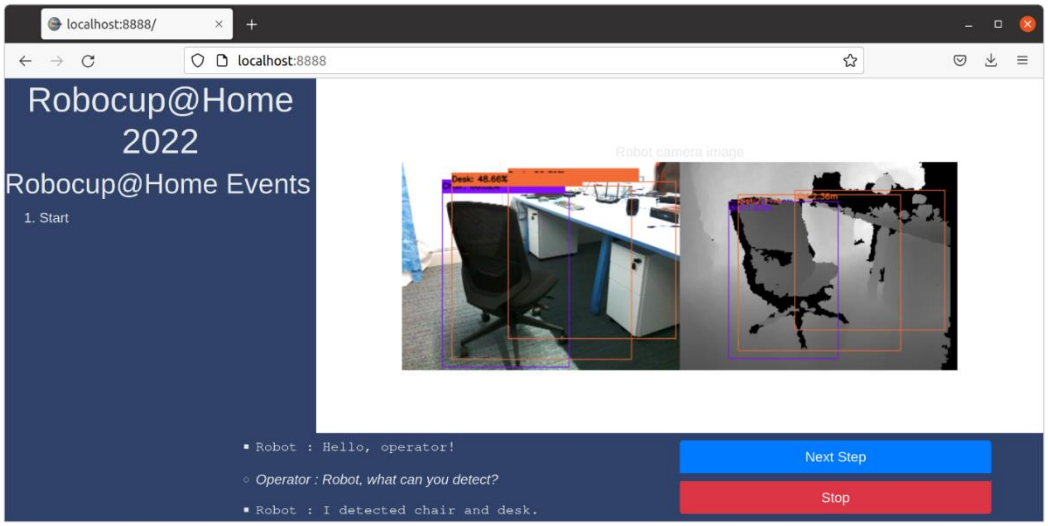

Figure 4. VizBox user interface. On the left side, it displays the steps in the competition. On the right side, it displays the robot's perception. On the bottom, it displays the communication between the robot and the operator.

2.4 Differences from expectation

The main difference of my actual tasks from what I expected is the computation power of the robot and the packages available on the robot. Most modern machine learning algorithms and applications require powerful computing resources such as GPUs and specialized packages such as PyTorch and TensorFlow. However, the Pepper robot only have a quad-core CPU at 1.91GHz without any GPU, and packages such as PyTorch are too large to be built into the robot. As a result, many state-of-the-art machine learning algorithms or deep learning models cannot be implemented on the robot because they will cause severe delay.

3. What I Have Learnt

3.1 Skills and knowledge I have learnt

Through this internship, I learnt a lot of knowledge in robotics and skills in Robot Operating System (ROS). For example, I learnt how to publish messages in ROS topics and subscribe to the messages in order to interact with the robot, as well as how to use rviz. I learnt how to use different devices on the robot such as the laser sensor, ultrasonic sensor, RGB camera, and 3D camera for different tasks. For example, the laser sensor and ultrasonic sensor can be used to detect obstacles. The RGB images from the RGB camera can be used to perform object detection while the 3D camera can be used to detect the distance of an object from the robot. These knowledge and skills would be useful to apply for jobs in the intersection of robotics and machine learning.

Moreover, I learnt how to implement machine learning algorithms with limited packages/libraries. Due to the constrained computing resources on the robot, many popular machine learning packages such as PyTorch, TensorFlow, scikit-learn cannot be built into the robot. Therefore, we cannot directly use machine learning models that are implemented on those packages. We primarily used OpenCV in Python

which is able to perform common computer vision tasks such as object detection and object tracking with pre-trained models. I have become very familiar with OpenCV due to the extensive use of it during my internship, which would be useful when seeking jobs in computer vision. When a method is not available in any package on the robot, I had to implement it from scratch. For example, we did not have pandas (to work with DataFrames) and webcolors (for conversion between RGB values to color names) on the robot at the early stage of my internship. I overcome the difficulty by manipulating data as lists instead of DataFrames and implementing a method to search for color names based on the closest RGB values in a table. Although it is less convenient than using the readily available methods, this experience has significantly improved my programming proficiency.

In addition, I have learnt how to migrate between Python 2 and Python 3. Although Python 3 is widely used in recent years, robotic applications that are implemented on older distributions of ROS might still require Python 2. The differences between Python 2 and Python 3 are not only in some syntaxes, but also in the available packages/libraries and how they work. For instance, when using the web framework tornado, we do not need to explicitly create event loop for a thread in Python 2, whereas we have to do this in Python 3.

3.2 How I used my knowledge from previous courses

During my internship, I have used a lot of knowledge and skills I learnt from my previous courses. For example, I learnt basic Linux commands from the course “Foundations of Computer Science”. This is very useful because it is more convenient to work with ROS in Linux than in other operating systems and I primarily used Linux system (specifically Ubuntu) in this internship. Although I did not take any course that teaches Python, I developed programming skills in Python through programming assignments in my previous courses, which enabled me to immediately start implementing the required functionalities without having to learn a new programming language. The knowledge in data structures and algorithm complexity analysis from the course “Algorithm and Data Structure Analysis” is very important to design an efficient algorithm as the processing power of the robot CPU is constrained. Moreover, I found the skills in Git and GitHub that I learnt from “Software Engineering and Project” very useful because the team has around 7 members working on different modules of the system (e.g., computer vision module, natural language processing module, navigation module, etc.), and we need to perform regular testing to make sure different modules can be integrated together. We use Git and GitHub for version control and integration between different modules. Last but not least, I learnt a wide range of machine learning algorithms in the course “Introduction to Statistical Machine Learning”. This has enabled me to design a suitable machine learning algorithm for a given problem and evaluate its performance using suitable evaluation metrics.

4. Advice for Other Interns

4.1 Useful knowledge for an internship

As every internship has different tasks and therefore requires different areas of knowledge, it would be difficult to include all the knowledge areas that are useful for every internship. Nevertheless, most internships in computer science will involve programming, most likely in a team of programmers. Therefore, familiarity with version control systems such as Git would be very useful. Also, it is essential to

be familiar with at least one programming language because it is usually easy to learn a new language if you already mastered one. Moreover, in real world, computing resources are usually constrained compared to the amount of data that needs to be handled. Therefore, good foundations in data structures and algorithms are required to design an efficient algorithm.

As for skills, since you would most likely work in a team, you need to be able to maintain good code style so that other people can understand you code easily. Also, you need to be comfortable with searching for solutions online when you encounter an unknown error/bug in your code, despite that the process might be tedious. Moreover, you need to be able to learn new knowledge and skills quickly because you may not have all the knowledge/skills required at the start of your internship. In addition, you would need some soft skills such as communication skills and teamwork skills to efficiently collaborate with your supervisor and colleagues.

4.2 How to prepare for an internship

To prepare for an internship, you can learn the knowledge and practice the skills mentioned earlier. For Git and GitHub skills, there are lots of tutorials online and you can practice by using it to manage your own projects or programming assignments. For programming languages, you can practice by solving programming problems (e.g., LeetCode problems) on a regular basis. You can use different programming languages to solve the same problem to better understand their differences. These practice problems can also help you get familiar with useful algorithms such as sorting algorithms. Depending on the kind of internship you are looking for, you might need to get familiar with the specific frameworks or packages that will be used in the project. For example, if the project is related to robotics, it would be useful to learn ROS. If the project is related to web development, you can learn Django or Flask (for Python). For data structures and algorithm analysis, you would need to know the commonly used data structures (e.g., trees, heaps) and algorithms (e.g., binary search), and experiences with more advanced algorithms such as dynamic programming would be a bonus. In addition, you can practice your communication and teamwork skills through group assignments/projects in your courses.

4.3 What to look for in an internship

Through an internship, you can gain a lot of experience and knowledge in the specific area (e.g., robotics, web development) of your internship. Your programming skills could improve significantly, especially in the language you use in your internship. You will also develop strong problem-solving skills because you are likely to encounter lots of unexpected problems during your internship. You will become more familiar with the industry that is relevant to your internship. For example, you can learn about state-of-the-art algorithms for the common problems in this industry, or the most popular frameworks for a specific type of software. You could also develop strong communication skills and teamwork skills, as well as build connections with potential employers, experts in the area, and peer students.