ENG7111

Industry Internship

Final report

Presented by:

Ching Yat Jason Wong

1. Introduction to IRL Crossing

In this internship, I am working for a laboratory named IRL Crossing, in which IRL stands for International Research Laboratory. It is a French Australian Laboratory for Humans-Autonomous Agents Teaming. IRL Crossing is a partner of CNRS, University of Adelaide, IMT Atlantique, University of South Australia, Flinders University and Naval Group. The most notable partnership I noticed is with CNRS, most of the engineers, researchers and even professor Cedric is from CNRS. The laboratory is relatively new, launched only in 2021.

IRL Crossing has many ongoing projects, and the one I worked on is only one of them. The projects are built upon 4 different thrusts: New Models to Understand and Anticipate Human Behaviour, New Algorithms for Energy-Efficient and Human-Based AI, New Paradigms for Autonomous Agents/Human Interaction and Understanding and New Solutions for the Management of Hybrid Teams.

New Models to Understand and Anticipate Human Behaviour
New Algorithms for Energy-Efficient and Human-Based AI
CROSSING
French Australian Laboratory for Humans-Autonomous Agents Teaming
New Paradigms for Autonomous Agents / Human Interaction and Understanding
New Solutions for the Management of Hybrid Teams

Figure 1, IRL crossing 4 thrusts

2. Tasks I carried out

The project that I worked on is titled "Human-Robot Interaction: Detect User Implicit Need of Help and Generate a Set of Assistive Actions". It seems complicated, but mainly it is the coding of a robot called pepper to perform tasks. The basic idea is to understand human command such as "Bring me a cup of water" using NLP (Natural Language Processing) and perform such action. In which there are some complicated processes such as understanding the command, locating the table, locating and detecting the cup of water, and bringing it to the user.

In order to keep the research relevant and up to industry standard, we are participating in a competition called RoboCup@Home. It is an international competition featuring many teams and laboratories from all over the world. We represent a team in France called RoboBreizh. The competition takes place every year in July, so it is expected that my work would be in the competition in 2023.

In the competition, a home environment would be present for the robot to navigate within. There are living room, dining room and kitchen, mimicking a real life home. In the environment, the competition host would give some command to the robot and scores the team based on the performance.

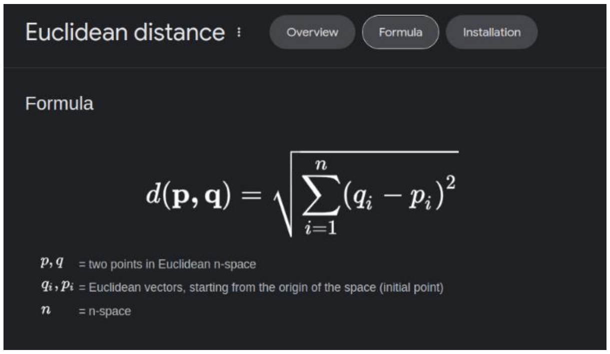
Below are some of the tasks I carried out during my Internship period.

2.1 color detection using k mean

My first task is color detection. In order to understand the command and perform task accordingly, color detection is a must. For example, when we command the robot to pick up a blue backpack and there are more than one backpack on the table with different color, the robot must understand the context of color for it to differentiate.

This is achieved by using the camera on the robot. First, perform object detection using the YCB dataset. Then using the detected object, crop the detected object as another image array. Keep in mind that in terms of cv2 (the library we are using), image is just an array of RGB values. RGB stands for red, green and blue, and the combination of three colors includes every color that human eyes can see. It comes in the format like (255,255,255). In the cropped image array, the pixels of the object have different RGB values because of lighting. This is where k mean is used to perform clustering. In a cropped image, usually there are two clusters of color, being the background color and the object color. The k mean clustering can determine which one is the dominant color in the picture and averaging the RGB value to generate one single RGB value for the object we want.

After that, we try to find the closest color according to the RGB value. While we can tell easily that (255,255,255) is white and (0,0,0) is black, the output value is normally something more like (192.5014, 18.9810 , 20.1021), value with decimal places. To solve this problem, I used the euclidean distance to calculate the distance between the output value and the common colors. The formula is below.



Euclidean distance

Formula

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

Figure 2, Formula of Euclidean distance

After finding the euclidean distance, we can determine which color the output RGB value is the closest to.

Notwithstanding, this is not the part which I struggle with. The part that requires the most work is the finding of a good csv file containing proper RGB value and the respective color. Most of the csv files that are available online is not something we want. They include some unwanted color names such as Bittersweet, Avocado and Amethyst. We want something simpler like green and blue. While we could only include basic color in the csv file, it does not work because in terms of euclidean distance, colors like light blue and light green are closer to white than blue and green. I struggled with the optimal csv file, but eventually I found one with good color names. After some editing work it becomes useable.

Below is the showcase of color detection on the robot.

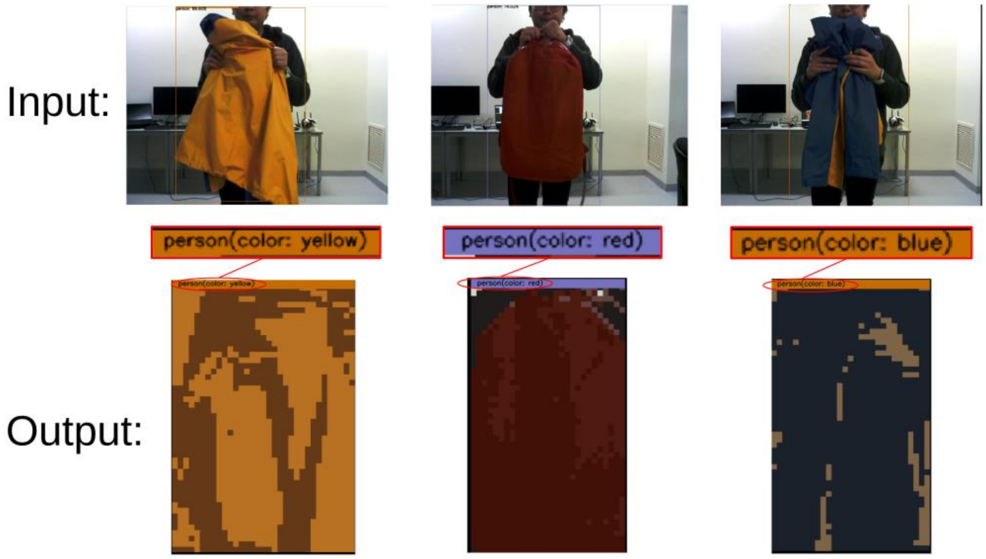


Figure 3, examples of color detection

2.2 Tracker

The second task that I worked on is the tracker. The human detection model is already available; however, the tracking of human is not present. The problem is that when the person in frame goes out of the camera range and goes back in, the robot does not know it is the same person or not. This part is essential because in the competition, there are commands like “Follow me to the living room”. In the navigation part, if the robot detects more than one person or the person goes out of camera, it is essential to track the person and determine which person to follow.

Because the video taken by the camera has different frames the first initial approach is to compare between frames. I have tried two different approaches to tracker the person in frame.

The first one is intersection over union (IoU). It is calculated by finding the overlap part of two bounding box, over by the area of union.

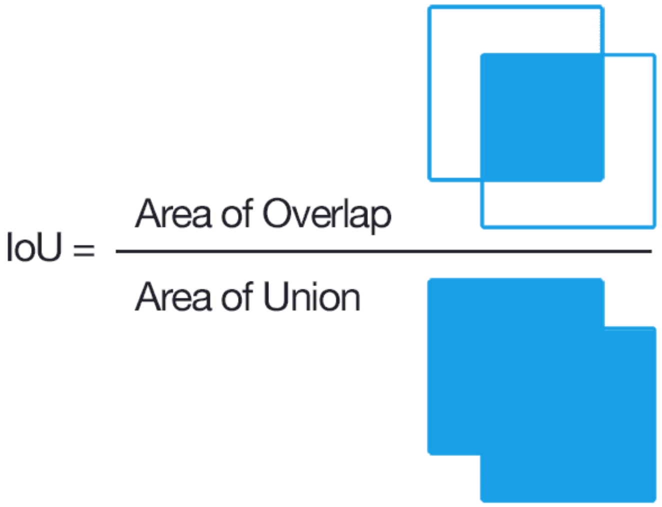


***Figure 4**, intersection over union (IoU)*

The number quantifies the degree of overlap between two bounding boxes, hence it could be used to determine the similarity between the bounding boxes between frames. This method performs best in most of the scenarios. If the frame person second is high enough, the movement by the person in frame will not be too drastic and the tracking works. When the person goes out of the frame, the next time the same person goes in frame would have the same bounding boxes.

However, this method is not perfect. If the person goes out of frame and goes in again with a different bounding box, this tracker would fail. For example, the person goes out left side and goes in right side, or the person goes in the frame far away from the camera, the bounding box would be different and IoU does not work.

This is where the second tracker comes in. I used my knowledge from my previous task and used it here. I implemented the color detection for each person in frame. This time I calculate the euclidean distance between this frame and the frame before to find out if they are the same person. I chose to use the RGB values directly to compare between frames instead of color names, as the color names are too general.

This tracker does not work perfectly either, as two people can wear clothes of similar color. This is where I took the average of IoU and RGB value, and generated a similarity number. In this way it can combine the benefits of both trackers.

This is the result of the tracker, video available on https://youtu.be/cl-GxaqC7uc

2.3 Difference from expectation

Before this internship, I would always assume image itself is a format. I did not know that image is just an array of RGB values. This changes the way I look at images. For example, I now know that cropping an image is just editing the array and making it the shape I want.

Apart from that, at the beginning of the internship I expect the robot would have no problem handling the code we put in. However this is not the case. The robot has limited computing power and some of the solution I come up with requires too much resource and is not feasible. Before the internship I did not care about the computational time, now I have an idea of what to do to reduce it. For instance np.power requires less time to run than traditional ** operator, and vectorization is better than looping.

3.1 skills or knowledge I learned

Overall, this internship has been a fruitful experience. It provided many things that I would not have encountered otherwise if not taken this course. I have never thought about being a team member and participating in robot competition. This is a unique experience.

In terms of skills, my coding skills have increased drastically. Before the internship, my coding skill only relies on school projects, or some small project on my own. This is the first time I take part in a big project, and the complexity of the code is on a different level. I spent a lot of the time reading through the documentation of cv2 and rospkg because I have never used these libraries before, but once I am familiar with them, I feel like I have improved significantly.

Not only my coding skills, but my mathematical skill has become better too. Some concepts like euclidean distance and IoU are not hard concept, but it is tricky to implement. By implementing on my own, I have improved my mathematical skills. The mathematical skill is actually more useful than I thought in this industry.

Other than that, I have learnt to comment my code. There are some incidents that the code is hard to read because the lack of documentation, and it really drags down the efficiency as I need to spend an entire afternoon just to understand it. Good documentation is needed to maintain code.

3.2 how I used my knowledge from other courses

The introduction course to programming course “Foundations of Computer Science” is vital in this internship. The course is mostly in Java, but it is extremely useful for some foundation and bash command usage. In the internship we mostly use Linux operation system and the bash command taught in the course comes in handy for many occasions.

The course “Introduction to Statistical Machine Learning” introduced me to many machine learning algorithms and k mean is one of them. The reason I could implement it quickly is because I have experience with k mean in the course. This course is very useful as basics, and I elaborate on that and learn a lot more new things about k mean in this internship.

4. Advice for Other Interns

4.1 useful knowledge for someone seeking to start an internship

One of the things I recommend getting familiar with is github. In most of the courses before, we usually use svn. Github and svn are both version control, but github is widely used by many companies to manage big projects. Some ideas such as pushing, pulling and branches are completely new to me. I recommend anyone who wants to start an internship to try playing a bit with github, and get used to the common commands. I spent some time understanding how branches work and what is a pull request.

Other than that, programming familiarity is a must. As interns need to read through old codes and change some of them, it is important to be familiar with the codes. Also have a basic idea of computational time. There are some courses that introduced us with big O, but in practical it is better to try out yourself using real life scenarios.

Same as any other industry, communication skills and team work skills are needed since we are working in a team. You should know when to consult your boss. Some basic things only requires a quick google search and you do not need to ask someone else.

4.2 How to prepare for an internship

It is very industry dependent, in the case of the lab I would suggest reading documentation of the libraries like cv2 and rospkg. The documentation is very thorough on the usage, reading through them makes you more familiar with the functions available and the ways to use them.

Also I would suggest watching tutorial on the packages. There are many videos available on youtube on each of the topics like object detection, color detection and nlp etc. The online resources are a great way to start as it could be more understandable and friendly to beginners than reading documentation.

Other than that, I think setting up a github account is a great way to prepare. Understanding github is very essential to internship and working in any project. Create your own small project and play with the commands a bit, It would make you more familiar with the version control system.

4.3 What to look for in an internship

I suggest get the best out of an internship, meaning look for every opportunity to learn something. In my opinion the most important aspect of an internship is to gain experience and learn something out of it. Do not be afraid of some package you have not seen before, take your time and get familiar with it. Some of the package like ros is used only on robots, it might not be useful in your future, but knowing more packages is good for you as many packages have similar function. Take every opportunity to learn in general when encountering new things.