ROBOT NAVIGATION USING OBJECT DETECTION - 2020

By Alex McLeod (19493178)

Curtin University – Computing
Supervisors: Nadith Pathirage and Wan-Quan Liu

Contents

Introduction

Artificial Intelligence is being used throughout the world and is only being used more as technology advances. This is seen through the various areas AI is being used in, such as with language recognition services like Siri and Alexa and through autonomous self- driving cars that drive by themselves. (Faggella, 2020). As a student in computing, gaining knowledge in Artificial Intelligence would be extremely beneficial when trying to find a job in the industry, as more employers are looking for people with experience in this new and evolving field. Artificial intelligence specialists being the 2nd most emerging role in 2022 as predicted by the World Economic Forum. (World Economic forum, 2018). Hence, I chose to complete a project on object detection so that I might gain the experience necessary to excel in the field and develop code that might benefit society.

Aim

The aim of my project was to use an object detection model to provide guided navigation to a simulated robot where it avoids objects that are detected in front of it. The final program should have the robot start by moving forwards and should use an object detection model to detect objects that appear on the robots camera as it moves forwards. When these objects are detected and are found to be at a small distance from the robot the robot should perform a 90-degree rotation to avoid them. The robot should then continue moving forwards until another object is detected at a close distance and it is made to rotate again. Only once the user presses the ‘q’ key for quit will the program end. Whilst this cycle occurs the user should be able to see the objects that are being detected in each frame.

Intended Personal Outcomes

Whilst working on my project this year one of my main goals was to develop the personal skills necessary to help me in completing an honours project in 2021, with the final goal of being able to acquire a job that I enjoy after my degree. These personal skills were outlined in the self-assessment that I performed at the start of the year.

One specific skill I wanted to develop as highlighted in my self-assessment was my communication skills. Communication is a skill that everyone needs if they are going to be successful in their career. Being an introverted person, I believe I need to develop this skill so that I can learn to display to others my capabilities and my ideas. Once results are produced it is through communication that you can show the importance of these results. This skill will help me in the final industry showcase and help me to become prepared for future work in my field. This year I planned on building this skill by regularly putting myself in situations where I could practice my communication. This would involve participating in regular group and one-to-one meetings with my supervisors to explain the work I am doing and the ideas I have. It would also involve taking any opportunity to present my work to others, allowing me to develop my public speaking skills.

Another skill in my self-assessment that I would like to develop by the end of the year is my time management skills. This will be important for my final honours year where I will have to manage my time wisely to ensure that I am making progress with my honours project, whilst still performing well in my other units and other non-university activities. There have been several times in the last few years where I have chosen to put less focus on particular tasks resulting in stress and work that is not of high quality. This year my goal was to allocate time to all my tasks ensuring that all were given adequate time.

Lastly, I wanted to develop my ability to work independently. This will be important for my final honours year as instead of someone telling me what I should be doing, it will be my responsibility to seek out my own honours project. This will involve communicating with a number of different people to find possible opportunities. Once I have found an honours project it will also be important that I am able to do my own independent research on topics necessary to solve the problems I encounter. I would develop this skill this year by seeking my own project and doing the research necessary to complete it.

Justification

Object detection is one of the most thriving areas of research in artificial intelligence with many technologies in today's world requiring object detection, making it an important area of study if we wont to create new technologies that can help and improve day to day tasks.

With the development of self-driving cars there has been a need for reliable and accurate object detection algorithms. In particular this includes algorithms that can detect objects such as other cars, pedestrians, trees and obstacles found on the road, allowing a car to react correctly depending on what and where it detects something, ensuring the safety of the driver within the vehicle. Through the development of convolutional neural networks (CNN's) for image classification and the use of object localisation algorithms for locating objects in an image, models have been created allowing for classification and detection of multiple objects within an image (Simhambhatla et al., 2019).

There have also been real time object detection algorithms used within cameras for security purposes. For example, algorithms can be used to detected weapons and suspicious items held by people within retail stores so that robbery and criminal acts can be reported. Other suspicious activity could also be reported such as luggage or bags that are found unattended at highly populated public areas or cars that are moving faster than the speed limit. Object detection could help to pick up suspicious acts such as these so that they are found at an early stage and can be dealt with accordingly by law enforcement. (Security and Safety Things, 2019).

Thus, as a computing student it is important that I complete this project in object detection so that I might help in the development of advanced algorithms that could help society in becoming more efficient and safer.

Note: This justification was previously submitted in my project plan

Background Research

Deep Learning and Convolutional neural networks (CNN's)

To begin working on the project research had to be done on the basics of deep learning. With the main focus being on Convolutional Neural Networks (CNN's). This would act as a refresher on important Artificial intelligence topics which would need to be understood for this task. CNN's being fundamental to understanding how image classification works in object detection. CNN's are deep learning models that are used in many computer vision fields. This is because they can be trained to identify different patterns within data, allowing them to then perform classification of an image based on these patterns. The basic idea is that a Convolutional neural network will first be trained. In the case that you want to classify images this will involve a number training images being input into the network, along with their corresponding labels which identify what the image should be classified as. E.g. a dog, cat, person, etc. Using these images and labels, the network will associate the patterns that it identifies with specific labels. Once trained the model can then be used to classify images that are input into the model. For any input image the model will output a label classifying the image. (Saha, 2018).

CNN's are made up of several complex layers that can be combined in different ways to suit the problem you are dealing with. One of the most important layers is the filtering layer. Each filtering layer consists of several filters. Each filter is a small matrix representing a particular pattern, the matrix scans over the image to check if that pattern is present in the image. It then produces a filtered image which represents how close that image is to having that pattern. Once we get to filtering layers further in the network more distinct patterns are identified. For example the first filtering layer may identify geometric patterns such as squares, corners, triangles, The next layer may identify more specific features such as feathers, eyes, hair and the final filtering layer may then identify cats, dogs, faces, etc. (Rohrer, 2016). Another important layer is the pooling layer. This layer usually occurs after the filtering layer. It will take each filtered image and identify the critical patterns within it, removing noise such as the background of the image and other parts of the image that might not be important for classification. They in turn reduce the size of the filtered image making it much easier to work with. The last layer in a CNN is the fully connected layer. This layer takes the final filtered images and decides what the final classification should be. Each filtered image will be associated with a label and give its vote on what the input image should be classified as. The label with the highest probability value will be considered the final classification of the image. (Saha, 2018).

The model is trained using a method called backpropagation. This involves tweaking the values of the model such as the values of the different filters so that they can classify the input image properly. Before you start training the model the values in the model such as the filter values are set to random numbers. The training images are then input into the model. Because each training image has a label associated with it, we can use this label to compare the expected value with the computed value to get an error value. When a large error value is computed it means that the computed value is far of the expected value. (|Error = expected value – computer value|). To reduce this error value, a method called gradient descent is used which involves tweaking values in the model such as the values of the filters to minimize the final error value. This process of inputting training images and performing gradient descent will continue until the error value is small and the model can accurately identify image. We want to make sure we don't over train this model as this can cause the model to memorise the training set rather then learn general patterns. By not overtraining the model will be able to identify new images outside the training set of images, by recognising similar patterns. (Rohrer, 2016).

Object Detection:

Research was also done on the different types of object detection models that are available. This would give us an idea of the possible types of object detection models that could be used for the project. Object detection involves taking an image and identifying any objects that are in that image, placing boxes around these objects and obtaining classification labels for each object identified. It is a combination of image classification which involves classifying an entire image and image localization which involves identifying where multiple objects are in an image. (Brownlee, 2019)

The first family of models that was looked at was Region Based Convolutional Neural Networks (R-CNN's). This family contains three models the R-CNN, the Fast R-CNN and the Faster R-CNN. These models involve finding possible areas where objects could be and then using a CNN to perform classification on each region to see if an object is present. As the names imply these models were improved over time to be faster, almost being able to achieve real time classification. They are generally very accurate. (Ardian Umam, 2017)

The other family of models that was researched was the You Only Look Once (YOLO) family. These models are very popular due to their ability to provide real-time object detection. Though they generally are not as accurate as the RCNN family of models they are a lot faster at performing object detection, sacrificing accuracy for speed. A lot of testing was done with these models as a similar model would need to be used for the project. We needed a fast model so that objects in front of robot could be quickly detected before a collision could occur. We looked at the code of these models and the code that used them to gain a better understanding as to how they work. Specifically, we looked at the YOLOv3 model from the website Machine learning mastery. (Brownlee, 2019)

Methodology

Implementing An Object Detection Model:

The first step was to implement an object detection model that could be used by the robot to detect objects from its camera. It is only once these objects have been detected that the robot can choose to avoid them. This implementation should take frames from a camera and enter them into a machine learning model. It should then extract the output from the model which includes coordinates of bounding boxes, labels of object detections and probabilities of these detections. It should place the bounding boxes around the objects in the frame and then place the names of the objects detected above these boxes along with the probabilities of the detections. This will allow us to then combine this code with the robot and capture its frames for object detection. By displaying each frame it will allow the user to see the detections that take place in real-time.

The type of object detection model we chose to use was a TensorFlow Lite model. These models are created using the TensorFlow library, which is an open source platform used to create machine learning models. The Lite models are designed so that they can be used on systems with low computational power, making them perfect for robots such as the Locobot robot that we used. The low computational power needed to run the model also saved us from having to set up a server to send data to and from the robot to perform object detection. Instead the object detection model can be run from the robot itself. The TensorFlow lite model that we used was created using ssd_mobile_v1 model and trained used the COCO dataset. The ssd_mobile_v1 is what is known as a single shot detector meaning that it can detect multiple different objects using a single input image. The model is able to detect all ninety objects found within the COCO dataset. This includes generic objects such as chairs, tables, cars, stop signs etc. (Object detection | TensorFlow Lite, n.d.).

To begin implementing this TensorFlow lite model we started by creating an Ubuntu 16.04 virtual machine. This Virtual machine allows us to run the Ubuntu operating system which is the same operating system at the Locobot robot. By using this machine it allows us to use the Locobot robot and run code on it. We also installed python 2.7 and TensorFlow 2.1.0 packages onto the machine. Python 2.7 being the programming languages that the robot runs with and TensorFlow 2.1.0 being the machine learning library that allows us to implement the TensorFlow lite model. (*Getting started with PyRobot · PyRobot*, n.d.) Using Python we first wrote code using the OpenCV library to capture each frame from my web cam, OpenCV being a python library built for computer vision tasks. Each frame was then resized to 300 by 300 pixels to be inserted into the TensorFlow lite model. Once inserted we were then able to extract the bounding box coordinates, labels and probabilities for each detection from the output. (Object detection | TensorFlow Lite, n.d.). Using the coordinates of each bounding box I was then able to use a function from OpenCV called rectangle which allowed me to draw the bounding boxes onto each frame. Using these coordinates I also placed the name of each detected object and its corresponding probability above each bounding box. I then redisplayed this image to the user with its detections using OpenCV's imShow function. This process is continually repeated so the user can see detections in real-time and only end's when the user presses the 'q' character on their keyboard. (*OpenCV-Python Tutorials*, 2013)

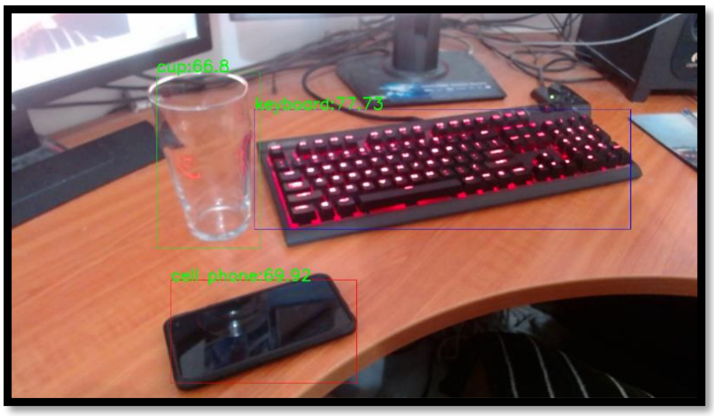


Figure 1. TensorFlow lite model in action using webcam

Creating the Simulation Environment:

The second step of the project was to then create a simulation environment where we could use an object detection model for navigation and object avoidance. This would involve spawning in a simulated Locobot robot and creating a simulation made up of different objects that the robot could interact with. The software we chose to use to create this simulated environment was called Gazebo. We chose as it allows you to easily create complex scenario's and is designed to allow you to test robot code. To customize our own environment we started by making a .world file. Within this file we can specify the different default gazebo objects we would like to place into the environment from Gazebo's database of objects. Within this file we can also customize the different objects by giving them a name, selecting their coordinates in the simulation, rotating them and customizing their physical properties such as whether they are static or not. This allows us to customize the environment appropriately for our application. (The Construct, 2018)

To test my final guided navigation code, I would need a simulation where objects are deliberately placed in front of the robot and would test its ability to avoid them. I started by placing three stop sign objects in front of the object. This would test whether the robot is able to detect multiple objects at the same time, detect which is closest and then rotate. I then placed a person object in the environment on the right of the three stop signs so that when the robot rotates the person will be in front of it. This will determine whether the robot is able to detect a different object and successfully rotate. More stop signs were also placed in the environment to make the robot continue moving in a rectangular path around the environment. This layout and expected path can be seen below:

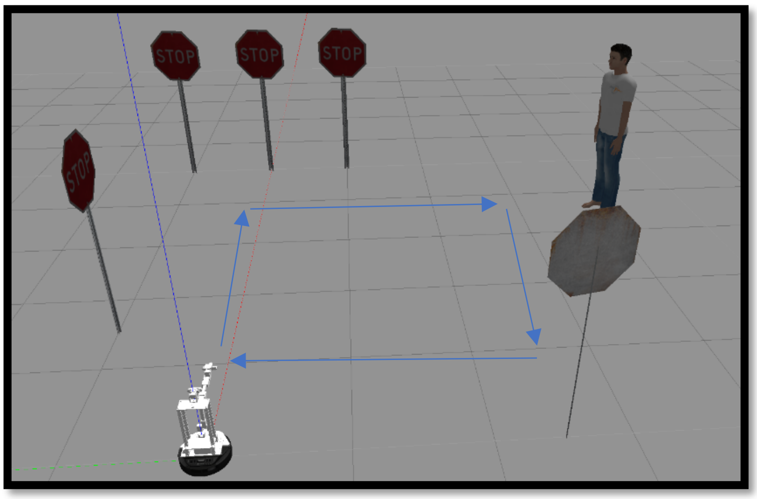


Figure 2. final simulation environment

Using this simulation environment, I also tested the basic functionality of the robot and created code that will perform this basic functionality. This was done with the help of second year advanced science students Dhirren and Khai who were working with this robot and had done a number of tests with this robot to test its functionality. They helped in performing basic set up for the robot so that it would properly spawn into the simulated environment. This environment being run by first activating the virtual environment and then running the command "roslaunch locobot_control main.launch use_base:=true use_sim:=true use_camera:=true" which would load the environment with the robot spawned within it. Functionality for the robot was written using the python package PyRobot which contains functions that can be used to perform basic actions. Code was then written to allow for basic movement such as moving forwards and rotations which would allow the robot to navigate around the environment and avoid objects. Code was also written for camera movement and to allow access to the robots camera so that frames could be captured from it and detections performed on each frame. (Getting started with PyRobot · PyRobot, n.d.).

Guided Navigation Of Robot:

The final step involved combining the TensorFlow lite implementation and the simulation environment to perform guided navigation where the robot would navigate around the simulation environment and avoid objects that are in front of it.

This involved first inserting each frame from the robot's camera into the TensorFlow lite model. Then for each frame the bounding boxes, labels and probabilities for each object detected were extracted from the output. For each bounding box, the x and y pixel points were taken from different parts of the box. The first point was taken from the top left of the box and then points were continually taken every 0.02 units to the right until the right side of the box was reached. It would then go down 0.02 units and start taking points from left to right again. This process of taking points from left to right of the image was repeated until the algorithm reached the bottom of the box. In this way pixel coordinates were taken periodically throughout the whole bounding box. Each point was then input into the PyRobot function pix_to_3dpt which used the robot's depth sensor to obtain the 3 coordinates of the pixels in the 3D simulation environment. (*Getting started with PyRobot · PyRobot*, n.d.). Using this 3D point and the distance formula ($d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$) the distance from the robot's camera to that particular point of the object was calculated. Once each distance was calculated for each point in the bounding box, the point that was closest to the object was extracted. Once the closest point of each bounding box was found we then cycled through each closest point to check if any of the objects were less than 1.5 metres away. If an object were less than 1.5 metres away a 90-degree rotation was initialised otherwise the robot would continue moving forwards capturing frames. This cycle of continually capturing frames, moving forwards and rotating would continue until the user enters the 'q' character for quit.

As well as the navigation of the robot being displayed to the user each frame was also displayed to the user. This would allow the user to see what objects were detected and when they were detected as being too close. For each object that was detected I implemented an algorithm such that the box surrounding it would be green with the name of the object placed above it along with its probability. I also displayed points within each box so that the user could see where the distance from each box was taken from. If none of the objects were detected as being within 1.5 metres, then green text was displayed stating "objects detected". In the case that an object was detected as being too close the bounding box was made to be a red colour and the text "objects close, rotating!" was displayed at the bottom of the frame.

Results and Discussion

As a final result a program was produced that provides guided navigation to a robot via object detection. This program allowed the robot to detect certain objects that were in its way and navigate around them. In our case it was able to detect a single stop sign, multiple signs, and a human model in front of it. The robot would move forwards by default and upon being within 1.5m of an object the robot would successfully rotate 90 degrees and avoid the objects. In our test the robot was able to successfully follow a square path avoiding the first three stop signs, avoiding the person, and avoiding the two separate signs. This cycle continued until the q character was pressed.

As seen through figure 5, the program would allow the user to see the robot as it moves within the simulation and avoid objects. Through a separate window the user can also see the point of view of the robot from the camera as it detects objects in real time. This display shows what the objects are that are being detected. It provides information on what the different objects are and the probability of the detection. In figure 3 we can see how it can detect multiple objects at once and detects how close they are based on the different points within the boxes. Green boxes showing that they have been detected but aren't close to the robot, so no rotation is needed. In figure 4 we can see how the program can successfully detect when a sign object is too close as it displays a red box around the sign and displays the text "Object Close Rotating". The robot then rotates following this detection. Through figure 6 we can see how it is also able to successfully detect a human object and avoid it, showing its ability to detect different objects. This is also useful as people are everywhere and will enable the robot to adapt to situations where humans are present. In modern object detection systems such as those that are present within autonomous vehicles human safety is the highest priority.

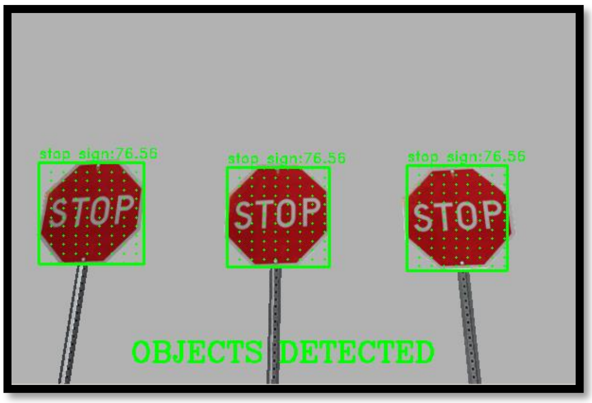


Figure 3. multiple stop sign detections

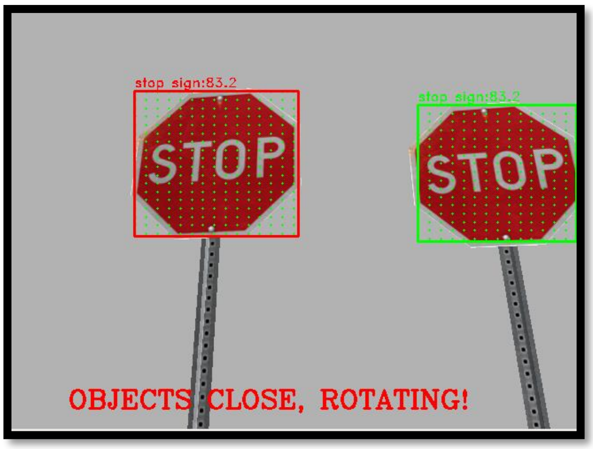
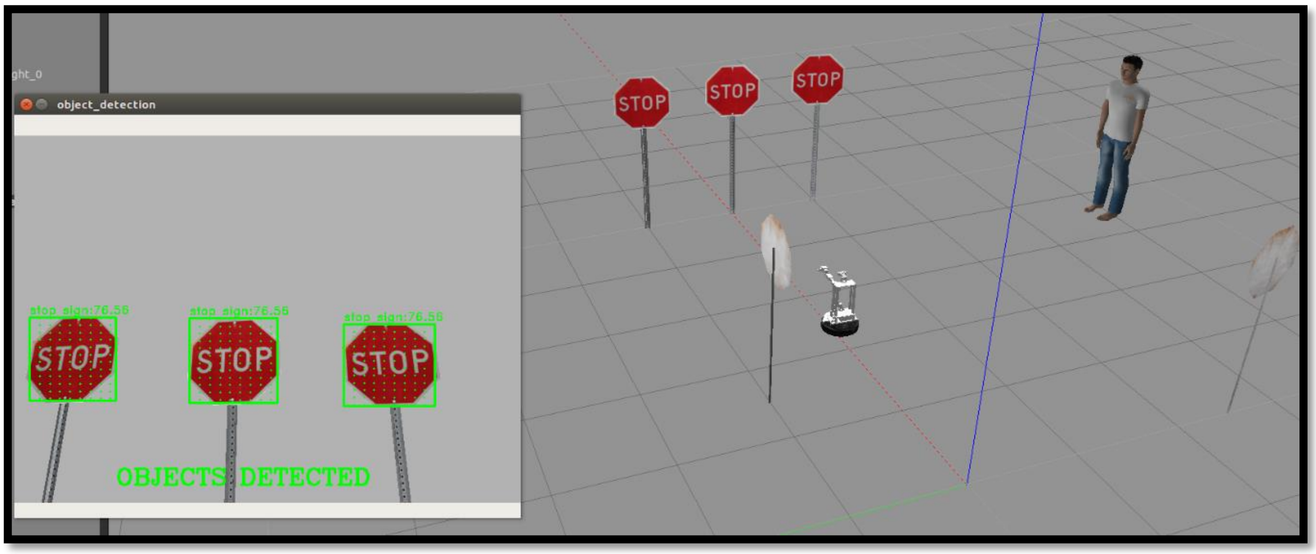


Figure 5. stop sign detected as too close

Figure 4. program in action



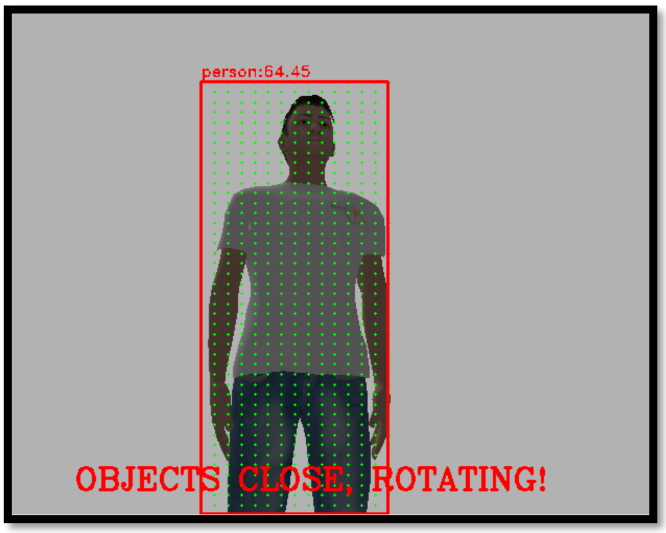


Figure 6. person detected as too close

Due to constraints caused by Covid-19 we did not have time to test the code with the physical Locobot robot and hence were forced to test the code only in the simulation environment. The simulation environment is very different to the real world as everything is made up of sharp edges and the background is a constant grey colour. Hence the accuracy of detections would be a lot less in the physical world due to more factors that need to be considered, such as different sized objects, varied lighting, and differing backgrounds. We would need to make a number of adjustments in the code to cater for these factors so that the robot can successfully avoid objects.

The default objects that Gazebo provides is very limited so we weren't able to test the robots ability to avoid any other objects which it has been trained on. For this reason its ability to successfully identify other items is unknown. We would have to learn how to create custom objects in the environment, train it detect new objects or test it with the physical Locobot robot with new objects.

This project helped in building a foundation for further work with the Locobot robot. By implementing this basic object detection it opens opportunities for more complex functionality of the robot. As once it is able to detect these object more functionality could be added to give the robot tasks depending on what it detects. Such as picking something up, moving something out of the way, etc. This project also helped me in building my knowledge of computer vision for future work in the field, whether that be for an honours project or industry work.

Personal Success and Learnings

I believe that I developed many skills this year as I had to overcome many challenges. With the outbreak of covid-19 at the start of this year I developed my ability to adapt to changes in my project. We were originally going to write code for the physical robot but due to covide-19 we were no longer able to come to campus to use it so instead we went completely virtual. This meant that I had to learn how to create a virtual environment and how to spawn a robot in this virtual environment. This added extra challenges to the project because before we could begin testing code, we had troubleshoot a number of issues to get the virtual environment working. We could also no longer hold face to face meetings and were forced to hold all meetings and communication online. So, I had to learn how to use a range of new online applications such as zoom and a team managing application called ClickUp to communicate with my supervisors and team members. Therefore I definitely learnt how to adapt to new situations which will be a valuable life skill as life always surprises you with unexpected challenges that you will need to overcome.

I also believe that I was able to improve my communication skills which is something that I mentioned that I needed to improve in my self-assessment. Being an introverted person, this skill was essential for me to build upon. To build this skill I was involved in as many meetings as possible so that I could learn to communicate my findings and update my supervisors. I also took opportunities to present my work where possible. I presented to my research group twice throughout the year and was a part of the industry showcase at the end of the year. Through these presentations I built upon my public speaking skills as I presented my work to several people and was provided with feedback on how to improve my presenting. Specifically I learnt to be concise and straight to the point rather then giving too much information which would provide less intrigue to listeners.

Another skill I aimed to build as mentioned in my self-assessment was my time management skills. Being in third year the majority of my units this year were of a very high difficultly so being able to manage my time between units and my project was of high importance. To mange my time properly I learnt to write down a schedule of things that I needed to do, which would include the different tasks and assessments that I needed to complete. By having this it gave me a better idea of the amount of time I had to finish my tasks so that I could complete them in time. For my project I also used a project management tool called ClickUp which allowed my supervisor to set tasks and sub-tasks for me to complete, with their corresponding due date. This also allowed me to keep track of tasks so that I could complete them in time. Learning to use a tool such as this will be very helpful for work in the future as many companies use similar tools for their projects. Hence now that I have learnt to use one it will help me in balancing workloads in the future.

Lastly another skill I wanted to build from my self-assessment was my ability to work independently. I developed this skill this year by seeking out my advanced science project. I firstly attended a small networking event at the start of the year for computing students. During this event I spoke with several different professors at Curtin which allowed me to develop my communication skills. I managed to speak with the professor I worked with last year Wan-Quan Liu and set up a meeting with him to discuss a project I could do with them this year. Through a number of further meetings, I was able to establish a plan for my project. Hence this gave me experience in how to seek out opportunities. Learning to seek out a project myself will be important for next year when I must find an honours project to work on. It will also be valuable when trying to seek out a job in the future as I will have to learn to apply for jobs and seek out opportunities to further my career.

Future Projects

The work I completed this year opens up a number of new projects. A project that could build upon my project could involve performing further tests with the TensorFlow lite model to test how well it can detect different objects. This could involve retraining the model on new data so new objects can be detected or using transfer learning to train the current model to detect new objects. Transfer learning would involve tweaking the current model instead of retraining a brand new one to detect new objects. This project could also involve testing the code on the physical Locobot robot at Curtin instead of the simulated robot. Because the real world is extremely different to the simulated environment a number of factors would need to taken into account leading to a number of tweaks to the current code for it to work.

In terms of new projects the work that the second year advanced science students and myself completed this year laid down the foundations for future projects with the robot simulation as well as the physical robot. This is mainly because we figured out how to create a virtual environment and how to spawn the simulated Locobot robot in the environment. This required a lot of trouble shooting and solving of errors. Now that we have solutions to these problems future projects with the robot will be easier as it will take less time to set up with our current knowledge. We have also learnt how to run code to perform basic functionality such as accessing the robots camera which will be essential in carrying out other projects on deep learning and computer vision. This could involve projects on object detection, classification, scene segmentation and a range of other projects which combine the functionality of the robot with deep learning and AI.

Conclusion

I believe that the Guided Navigation Using Object Detection project which took place in 2020 was a success. Though there were several complications due to covid-19 a final simulated program was produced which allowed the robot to successfully navigate and avoid objects that it detected. This project gave me a great foundation to continue learning about object detection and other work in deep learning. The project was also beneficial as it provided the necessary experience and knowledge for a possible honours project. It gave me experience in seeking out, planning, and developing a project. These skills will be essential when working towards an honours project in 2021 or any future work in the field. I was also able to develop several personal skills in communication, teamwork, time management and independence which will also be valuable in my professional career and my personal life.

Note: My supervisor report has been submitted separately via email by my supervisor Wan-Quan Liu.

Project Deliverables

The following is a small snippet of code that was used to determine whether a rotation should be initialised to avoid objects. It contains an algorithm that finds the distance away each object is, compares the distances of each object and decides whether any of the objects are too close. Source code, demo videos and photos can be found in the Final ePortfolio.

Sample Code:

```
# purpose: module for check the the distance of detected objects
def checkObjectDistance(boxes, img, bot):
    rotate = False
    isObjectClose = []
    boxIndex = 0
    # for each detected object check the distance from the robot
    for box in boxes:
        print("Box: ", boxIndex)
        #starting default value
        closestDist = -1
        ycoord = box[0] #top
        xcoord = box[1] #left
        bottom = box[2]
        right = box[3]
        while ycoord <= bottom:
            xcoord = box[1] # rest to the left
            while xcoord <= right:

                scaledX = int(round(640 * xcoord, 0))
                scaledY = int(round(480 * ycoord, 0))
                cv2.circle(img, (scaledX, scaledY), 1, (0, 255, 0), -1)
                print("x value: ", scaledX, "y value: ", scaledY)
                if((scaledX < 640 and scaledX > 0) and (scaledY < 480 and scaledY > 0)):
                    pt, colour = bot.camera.pix_to_3dpt(scaledY, scaledX, in_cam=True)
                    if((math.isnan(pt[0][0]) == False) and (math.isnan(pt[0][1]) == False) and
(math.isnan(pt[0][2]) == False)):
                        currentDist = calcDistance(pt[0][0], pt[0][1], pt[0][2])
                        if (closestDist == -1) or (currentDist < closestDist):
                            closestDist = currentDist
                xcoord = xcoord + 0.02
            ycoord = ycoord + 0.02

        print("Closest Distance: ", closestDist)

        if (closestDist < 0.0018 and closestDist > 0):
            rotate = True
            isObjectClose.append(True)
        else:
            isObjectClose.append(False)
        boxIndex = boxIndex + 1
    return rotate, isObjectClose
```

References

Brownlee, J. (2019, July 5). A Gentle Introduction to Object Recognition With Deep Learning. Machine Learning Mastery. https://machinelearningmastery.com/object-recognition-with-deep-learning/

Brownlee, J. (2019, May 26). How to Perform Object Detection With YOLOv3 in Keras. Machine Learning Mastery. https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/

Getting started with PyRobot · PyRobot. (n.d.). Pyrobot.org. Retrieved November 19, 2020, from https://pyrobot.org/docs/overview.html

(*Getting started with PyRobot · PyRobot*, n.d.) *LoCoBot (PyRobot)*. (n.d.). Www.Trossenrobotics.com. Retrieved November 19, 2020, from https://www.trossenrobotics.com/locobot-pyrobot-ros-rover.aspx

Dickson, B. (2019, January 14). *What is computer vision?* TechTalks. https://bdtechtalks.com/2019/01/14/what-is-computer-vision/#:~:text=Applications%20of%20computer%20vision&text=The%20importance%20of%20computer%20vision

Object detection | TensorFlow Lite. (n.d.). TensorFlow. https://www.tensorflow.org/lite/models/object_detection/overview

Faggella, D. (2020, March 14). Computer Vision Applications - Shopping, Driving, and More. Emerj. https://emerj.com/ai-sector-overviews/computer-vision-applications-shopping-driving-and-more/#:~:text=Computer%20vision%2C%20an%20AI%20technology

World Economic Forum (2018). *The Future of Jobs 2018.* Retrieved from http://reports.weforum.org/future-of-jobs-2018/shareable-infographics/

Rohrer, B. (2016). How Convolutional Neural Networks work. In *YouTube*. https://www.youtube.com/watch?v=FmpDIaiMIeA

Saha, S. (2018, December 15). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. Towards Data Science; Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Ardian Umam. (2017). 1. Introduction to How Faster R-CNN, Fast R-CNN and R-CNN Works [YouTube Video]. In YouTube. https://www.youtube.com/watch?v=v5bFVbQvFRk

OpenCV-Python Tutorials — OpenCV-Python Tutorials 1 documentation. (2013). Readthedocs.Io. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

The Construct. (2018). [Gazebo in 5 mins] 001 - How To Launch Your First Gazebo World Using ROS [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=qi2A32WgRqI