Diploma in

Robotics & Mechatronics

Final Year Project Report

Project Title:

Outreach Robot

Project Supervisor:

Dr. Edwin Foo

Project Student:

Tan Yi Liang

182947S

Abstract

Deep learning approaches have been proven in the last decade to be able to outperform traditional machine learning techniques in several areas. Among the many approaches deep learning has to offer, Computer Vision is the most prominent case. Deep learning for Computer Vision is one of the futures of technology, even being utilized in the medical and military industry. Therefore, it is important to encourage and push students to take up artificial intelligence projects to understand how it operates, so that they can work with it in their futures.

This report paper will detail the project undertaken during the 12 – week Final Year Project from 1st June 2020 to 22nd August 2020. The project revolves around a 'Scissors – Paper – Stone' game, with the main concept being deep learning for computer vision, coded in the Python language. An account of the hardware and software used will be given, followed by the project scope and the project implementation. Finally, a brief overview of future directions with the project will be given, alongside with the potential improvements, the problems and challenges potentially involved therein.

Table of Contents

Introduction

Deep learning is a branch of machine learning approaches focused on artificial neural networks, alongside with feature learning, a collection of techniques allowing a program to automatically discover the patterns needed to detect features or to classify the source data. Architectures such as deep neural networks and convolutional neural networks are essential to fields such as computer vision, speech and audio recognition, which have been proven that the results achieved can equal, and in some cases, even far exceed the human's capabilities.

The core of this project is deep learning for computer vision. Computer vision is an artificial intelligence field which trains computers to analyse and comprehend digital elements, such as videos or images. It tries to identify and automate processes which can be carried out by human vision, such as object detection and facial expression detection.

Due to the potential and the rising demand of both deep learning and computer vision, it is important that schools start introducing modules related to artificial intelligence to develop interest in students to further advance and surpass the current technology. An effective way to do this would be to create an interactive software which allows users to experience what AI can achieve.

The ultimate goal for this project would be to incorporate the program onto an autonomous robot. This robot will be sent for expeditions around schools to let students play and interact with it. With this in mind, different applications that is both interesting and relevant to AI must be integrated into the game to attract students.

Objectives

The aim of this project is to modify and improve the program to deliver a more engaging and interactive game. The current program only consists of the game, and is not attractive enough to entice people to play. Thus, the project objectives for the 12 weeks will be to:

- Improve the accuracy of the user's gesture detection.
- Introduce object detection to the program.
- Introduce facial expression recognition to the program.
- Introduce text – to – speech elements to the program.
- Improve the Graphics User Interface (GUI) to be more engageable, interactive and fun.
- Optimize game code for faster start-up and detection.

Hardware

Since the end goal of this project will be to transfer the program onto an autonomous robot, it is important that the hardware not only requires little wattage, small in size, but to also be powerful enough to ensure the program runs smoothly without any hitches or crashes.

Nvidia Jetson Xavier NX

The Jetson Xavier is a powerful computer designed for deep learning purposes. It has an inbuilt GPU, a 6 – core CPU, as well as 8 GB of integrated RAM. This computer was chosen for its portability, as well as its low power requirement at a minimum of 10W. At a size of 70 mm x 45 mm, this compact computer would be optimal to place onto the robot.

Figure 1 - Jetson Xavier NX

Orbbec Astra Embedded S

This camera is selected for its small size (68.6mm x 22.35mm x 14.8mm), as well as its support for both depth and colour stream. The colour stream will be shown to the user, and the depth stream will work in the background to capture the user's gesture. This camera can display 30 frames per seconds (FPS) for the colour stream, and more importantly, up to 60 FPS for the depth stream. This will be an important factor in increasing the accuracy of the gesture detection, which will be touched on, on the later parts of this report. However, because this camera has its own processor inside, the temperature of the camera gets hot after some time, especially in a warm environment. This may lead to the camera displaying lesser FPS then it was programmed to.

Figure 3 - Orbbec Astra Embedded S

Figure 2 - Depth Stream

Figure 4 - Color Stream

Software

To fulfil the project objectives, the following software and modules will be used. A brief description of each software will be given.

TensorFlow 1.15.2

TensorFlow is an open-source platform for machine learning. It contains a large variety of tools, libraries and community resources that is suitable for deep learning.

Figure 5 - Tensorflow Icon

Keras 2.2.4

Keras is a neural network library built on top of TensorFlow. It is designed to be fast and easy to use. Keras is a high-level Application Programming Interface (API) used to train the Convolutional Neural Network model for the detection of the user's gesture.

Figure 6 - Keras Icon

OpenCV (Open Source Computer Vision Library) Contrib 4.1.2

OpenCV is a machine learning library designed to provide a platform for computer vision applications, namely image processing. It will be used for many functions, the more notable being reading and writing images, facial expression recognition, and object detection.

Figure 7 - OpenCV Icon

PyGame 1.9.6

PyGame is a module designed for creating video games in the Python language. PyGame will be used to create and design the Graphical User Interface (GUI) for the game program.

Figure 8 - PyGame Icon

The following software, while important for the program, is not as widely used as the software above.

OpenNI 2.3.0.63

Used to start the depth stream.

gTTS

Used for text – to – speech functions.

NumPy 1.18.5

Used to convert images to matrices and arrays.

Project scope

Deliverables

The final program uses a variety of functions to increase the interactivity of the game. The following are the different processes in the program.

1) Upon start up, the machine will scan for the presence of a person. Should there be a person within range of the camera, the program will then move on to the facial expression recognition function. Otherwise, the program will continue to run until the condition is met.

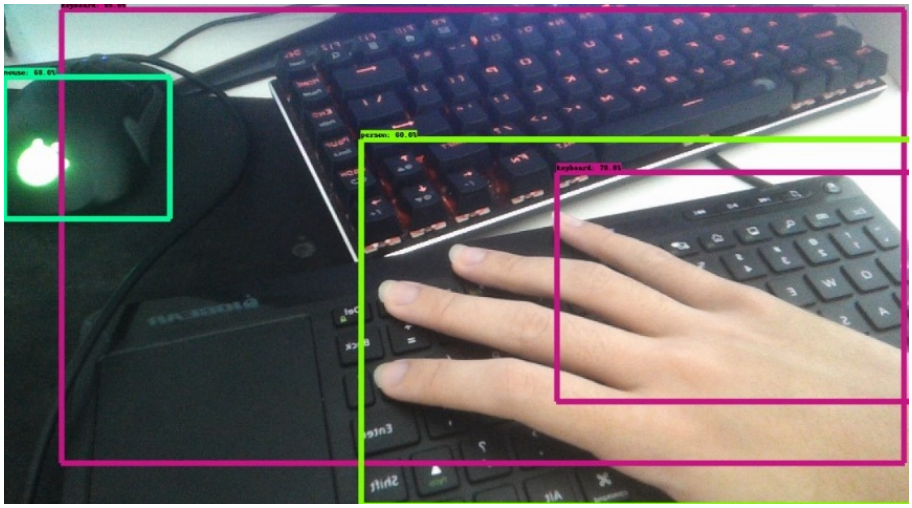


Figure 9 - Object Detection Example

2) Depending on the user's facial expression, the program will display a screen and initiate the text – to – speech function, inviting the user to play the game.
   2a) If user selects option 'Sure!' – go to 3)
   2b) If user selects option 'No thanks.' – go to 10)

Figure 10 - Starting Screen

3) The program will then open the main menu

Figure 11 - Main Menu

4) Upon pressing option 'Play', the program will move to a screen where the user selects a game mode.

Figure 12 - Game mode Selection

5) Once a game mode is selected, the instruction screen will appear as shown in Figure 11

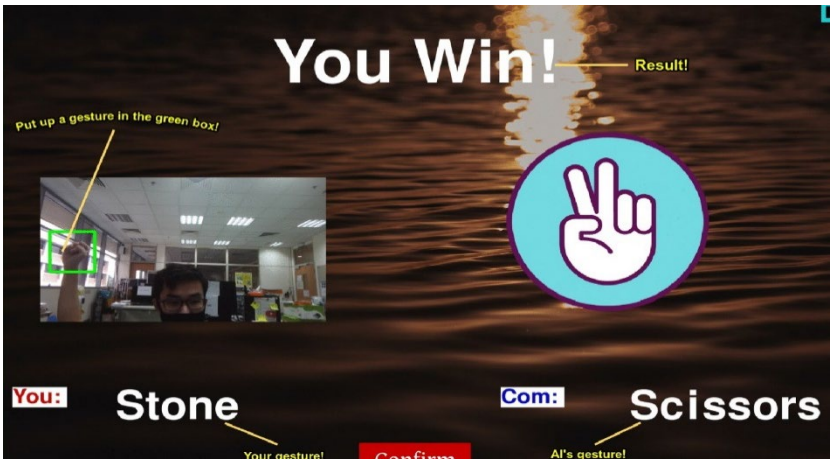


Figure 13 - Instruction Screen

6) The program will wait for the user to press the 'Confirm' button. After the confirm button is pressed, the program then moves on to countdown.

7) Once the countdown is done, the image within the fixed green box shown in the video frame will be cropped, and the converted image is then sent to the CNN model for prediction. The computer will then display a gesture (depending on the game mode selected). After 3 matches, the program proceeds to the end screen.

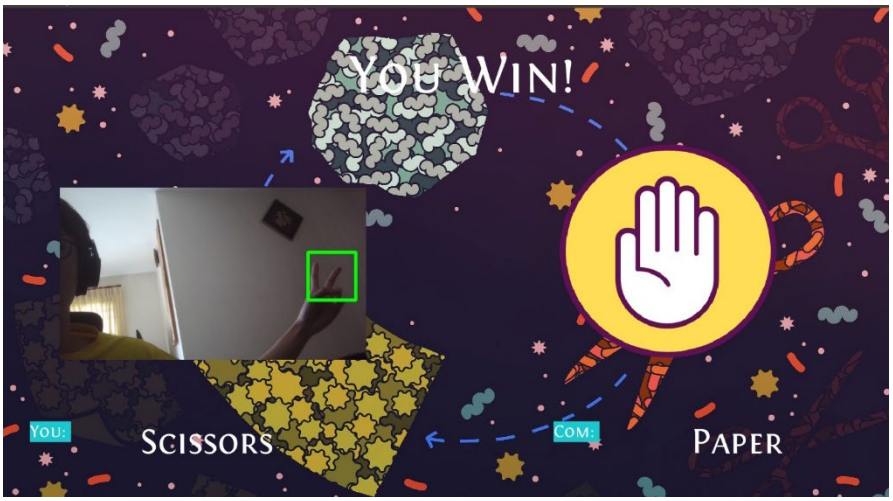


Figure 14 - Game Screen

8) At the end of the game, various options will be shown.
   7a) If option ‘Main Menu’ is selected – go to 3)
   7b) If option ‘Choose mode’ is selected – go to 4)
   7c) If option ‘Play again’ is selected – go to 6)
   7d) If option ‘Quit’ is selected – go to 10)

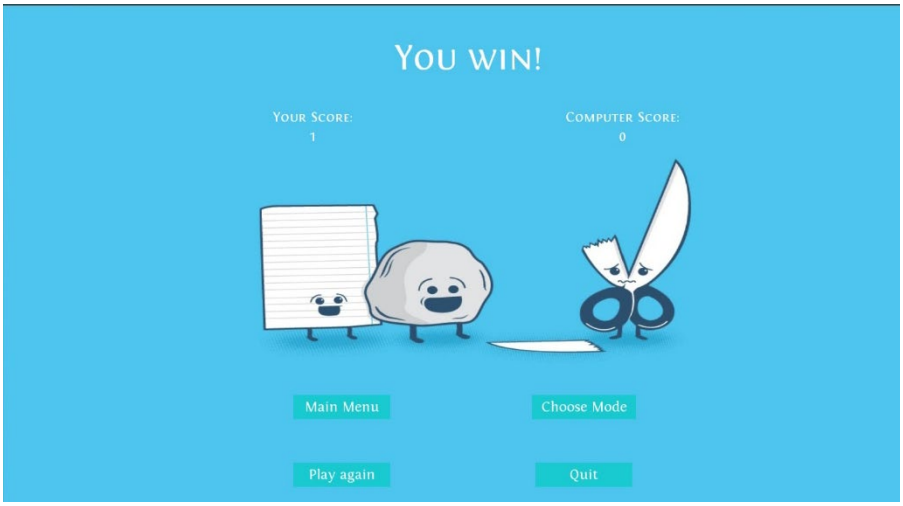


Figure 15 - Result Screen

9) Program will display a screen to invite the user to experience the AI’s capabilities.
   10a) If option ‘Sure’ is selected – go to 9)
   10b) If option ‘No thanks.’ Is selected – go to 10)

Figure 16 – Before Quitting

10) The object detection program will start up, showing the user the live camera feed, as well as the objects it can detect, along with the accuracy.

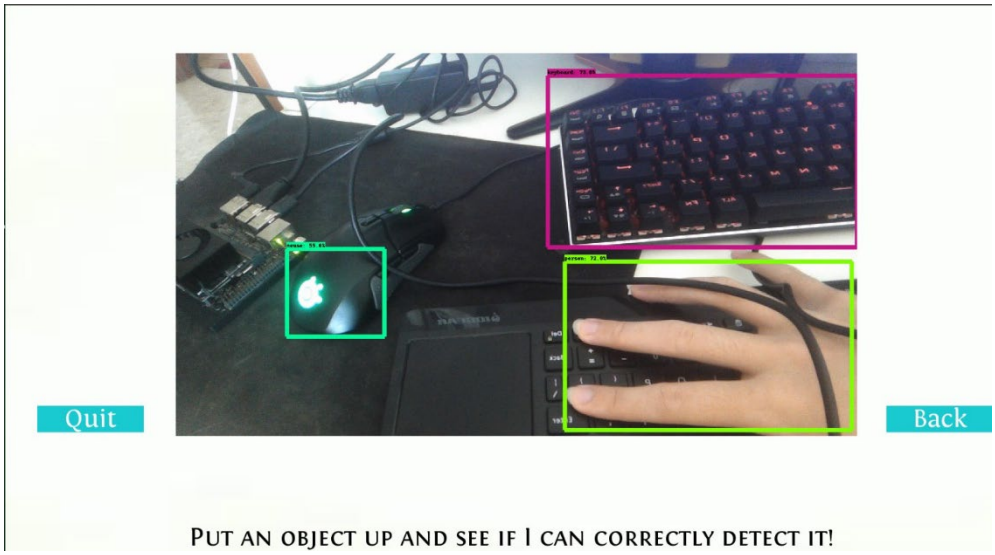


Figure 17 - Object Detection Minigame

11) Program shows a screen and quits after 5 seconds.

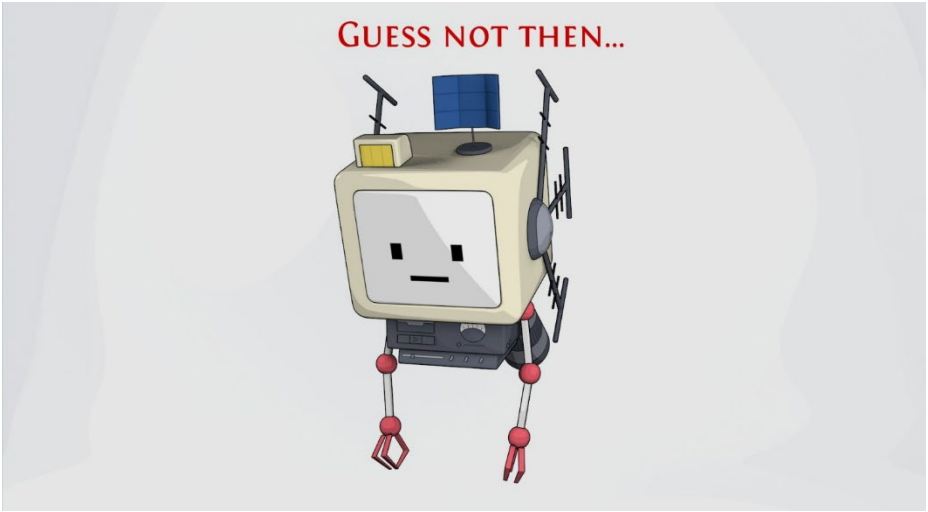
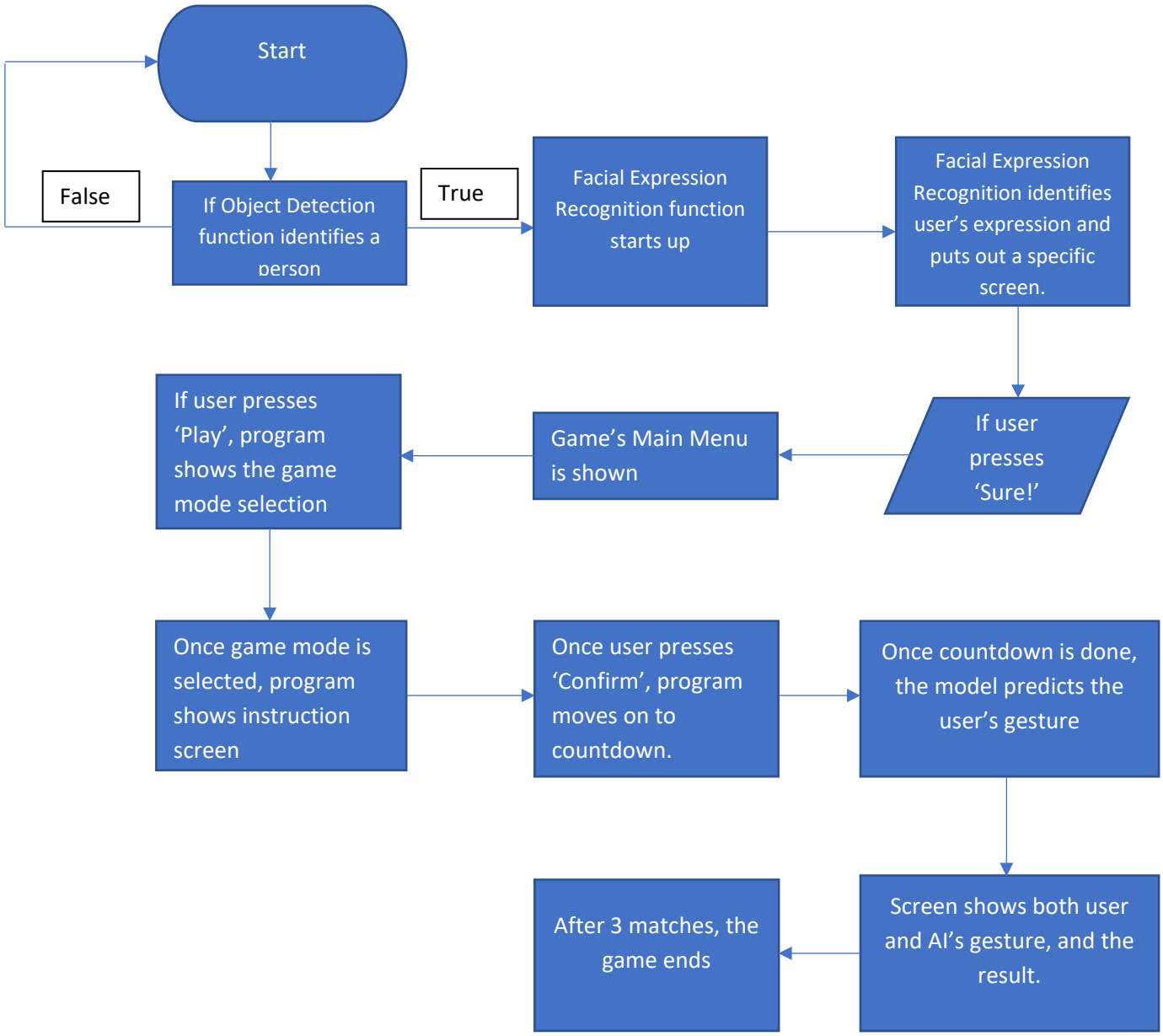


Figure 18 - Quit Screen

Gantt Chart

No.	Activity	1	2	3	4	5	6	7	8	9	10	11	12
1	**Knowledge Acquisition**												
	Python3 programming												
	Basics of Convolution Neural Networks (CNN)												
	Tensorflow and Keras												
	OpenNI camera programming												
2	**Setup**												
	Setup Jetson nano												
	Install necessary modules												
3	**Programming**												
	Create program to crop and capture gestures with the depth camera												
	Create program to predict gestures with the depth camera												
	Create program to Gaussian threshold captured images												
4	**Collecting and Training data**												
	Capture picture of the gestures												
	Augment gestures												
	Train image into a model												
5	**Game Programming**												
	Implement the depth camera prediction code												
	Improve and rewrite parts of the program												
6	**Facial expression recognition (FER)**												
	Gather data for a database												
	Train the model												
	Create program to zone in on user's face												
	Create program to detect user's facial expression												
7	**Object detection (OD)**												
	Find and select pretrained model												
	Create a program to detect objects												
8	**Incorporating functions into main program**												
	Insert FER and OD into main program												
9	**Final testing and optimization**												
	Finalize all programs												
	Planned												
	Actual												

Project Implementation



Software Development

Implementing the depth stream

To increase the detection accuracy of the user's gesture, the new Astra Embedded S camera is introduced to replace the HD webcam. Prior to this introduction, the methodology of detecting the user's gesture using only the colour stream from the webcam had many steps and factors that reduces the detection accuracy.

These are the factors:

1) The user had to scan the colour of his hand.

   The idea behind this process was to take the average pixel values of the user's skin colour to create a more consistent threshold. However, this also meant that external factors such as lighting and the consistency of the user's skin colour could potentially drastically reduce the detection accuracy.

Figure 19 - Original Screen

2) User's gesture is captured, processed into a binary image with the user's skin colour being depicted as white. The problems faced in the first step carries on to this step as well. If the user's skin colour was not properly scanned, the image of the gesture may be left with holes. Passers-by with a similar skin tone could also be captured, bloating up the binary image, and thus reducing detection accuracy.

Figure 21 - Original Detection

Figure 20 - Original Thresholding

Figure 22 - Best to Worst Image

3) The binary image is fed into the CNN model to predict the gesture.

With the problems introduced in the earlier steps, the clarity of the image sent may not be recognizable by the CNN model anymore. Although the CNN model had a 90+% accuracy, due to the many factors mentioned, the accuracy may be drastically reduced.

With the introduction of the new camera, along with the support of the depth stream, the problems faced are all resolved. The depth stream is only affected by distance, thus factors such as lighting will not be relevant. Since the user's gesture will be the closest to the camera while playing the game, the image captured is clear, with no bloat or noise. The depth stream also supports 60 Frames – per – Second (FPS), thus even in the case whereby the user switches gestures at the last moment, the image is still clearly captured.

Figure 23 - Depth Stream

Figure 24 - Depth Stream Crop

Gaussian Thresholding

Gaussian thresholding is a branch of adaptive thresholding methods. Applying the gaussian thresholding on the captured image results in the image being edge-traced. Since the three different gestures are not identical, it will be easier for the trained CNN model to recognize the user's gesture.

Figure 25 - Gaussian Thresholding

Adaptive Gaussian thresholding is best used when there are different lighting, or in this case, different depth conditions. The threshold value is chosen from a weighted sum of the neightbourhood values. It is able to edge-trace the object nearest to the camera, which should be the user's gesture.

Edge-tracing also provides a clearer distinction between the gestures. Since the three different gestures are not identical, combining that with the accuracy of the depth camera, the gaussian thresholding will prove to be easier for the CNN model to recognize the user's gesture.

The code snippet below applies a gaussian threshold filter to all images in the specified path.

Retraining of the CNN model

After the switch to the depth camera for image detection, the model and images used by the previous students were unusable. Thus, there was a need to gather new images and train the new deep learning network to recognise the images.

To re train the deep learning model (i.e. CNN model) , there must be a dataset that contain both the training and validation images. The training images were scrapped from the internet as well as taken from all the FYP student. In order to build up a bigger database for image training, The number of training images were enlarge by using image augmenting methods in the Keras package as shown in this code found in annex snippet A.

The augmenting process does a horizontal rotation, as well as randomly rotating the image less than 90 degrees. After augmenting, we have 3000 images of each gesture. There are 300 validation images of each gesture. They are randomly taken from the set of training images, then augmented again. The next step will be to apply the gaussian threshold filter onto every image, and then finally training it.

Currently, the model has some limitations. The model is able to predict gestures up to 60 – 70cm from the camera because the training images were all taken from around that distance. It is theoretically possible to train the model to predict the gestures at an even further distance, however, the accuracy will likely drop to 70% and lower.

Introduction of the object detection function

The object detection function will serve to be a showcase of the potential and capabilities of Deep learning with Computer Vision. As the ultimate goal for this project is to incorporate the program onto an autonomous robot, it is important that the robot is able to recognize people. This function will be able to do just that, as well as delivering a more engaging and interactive program.

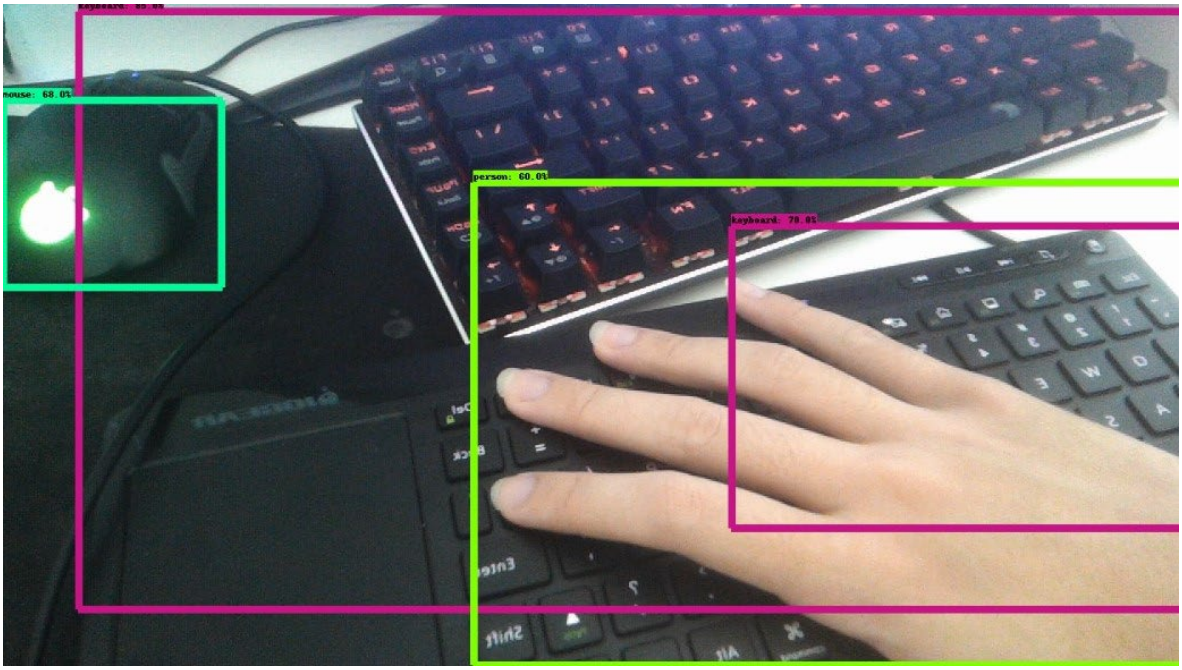


Figure 26 - Object Detection

The COCO SSD Tensorflow model used is only able to detect 100 objects – most of which being everyday household items such as TVs, cell phones and bottles. It is possible to add more identifiable objects through transfer learning or re-training the model.

Introduction of the facial expression recognition function

This function allows the program to respond to the user's mood.

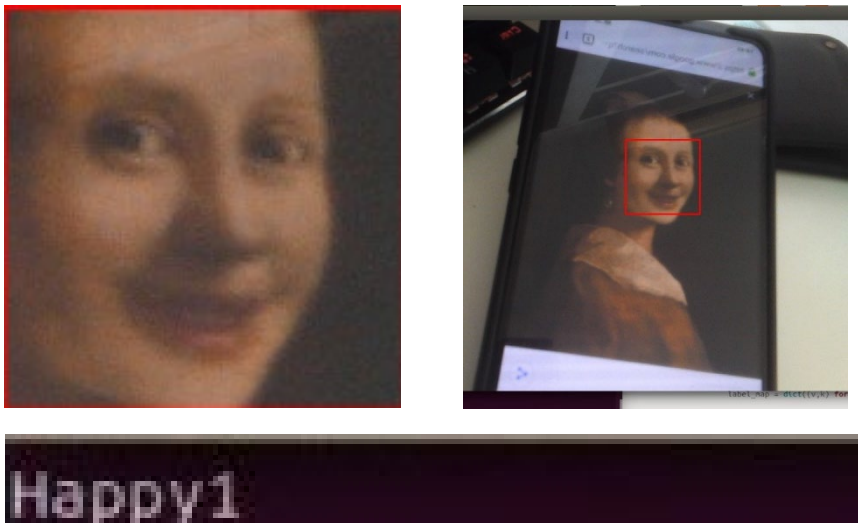


Figure 27 - Facial Expression Recognition

Using the 'face_recognition' python module, the program is able to recognize human faces. The program then zones in onto the user's face, crops it, converts it to grayscale and finally sending it to the CNN model for prediction. The model currently supports 7 emotions – angry, sad, happy, neutral, surprised, fear, and disgust. It is possible to add more identifiable emotions by either transfer learning, or by re-training the model.

Improving the attractiveness of the game

The original GUI is unintuitive. The buttons are not responsive, and requires the user to click it multiple times for it to register. A quick fix was implemented by disabling the music files. Upon deactivation, the buttons not only became much more responsive, but the overall program has also been running faster.

The background images were bland, low-quality, and unrelated to the game. Thus, new images had to be sourced from various websites. Below is a comparison between the old and new background.

Figure 29 - Original Main Menu

Figure 28 - Main Menu

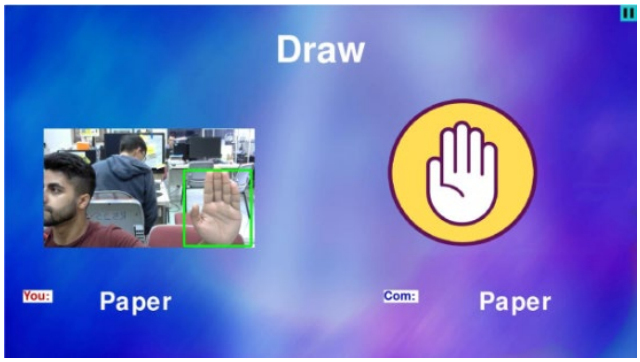


Figure 33 - Original Game Screen

Figure 32 - Game Screen

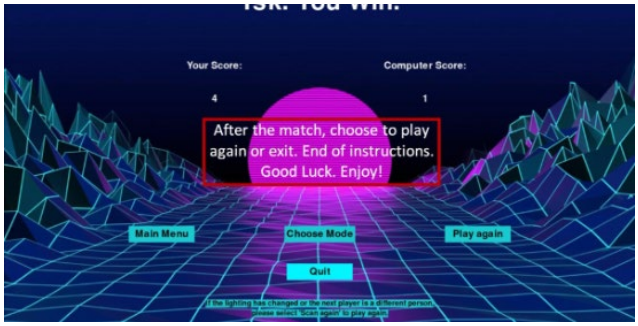


Figure 31 - Original End Screen

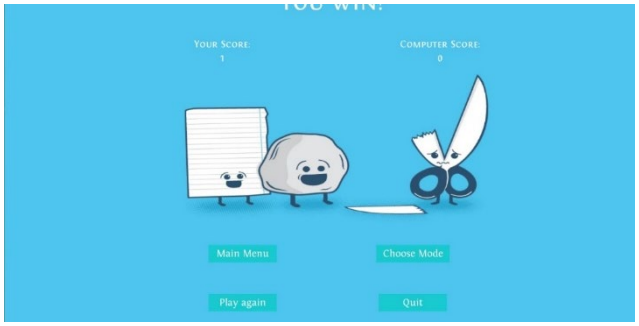


Figure 30 - End Screen

Miscellaneous work

- Text – to – Speech with the gTTS module.
- Setup the Jetson Nano (Currently unused)
- Setup the Jetson Xavier NX
- Various bug fixes and optimizations.

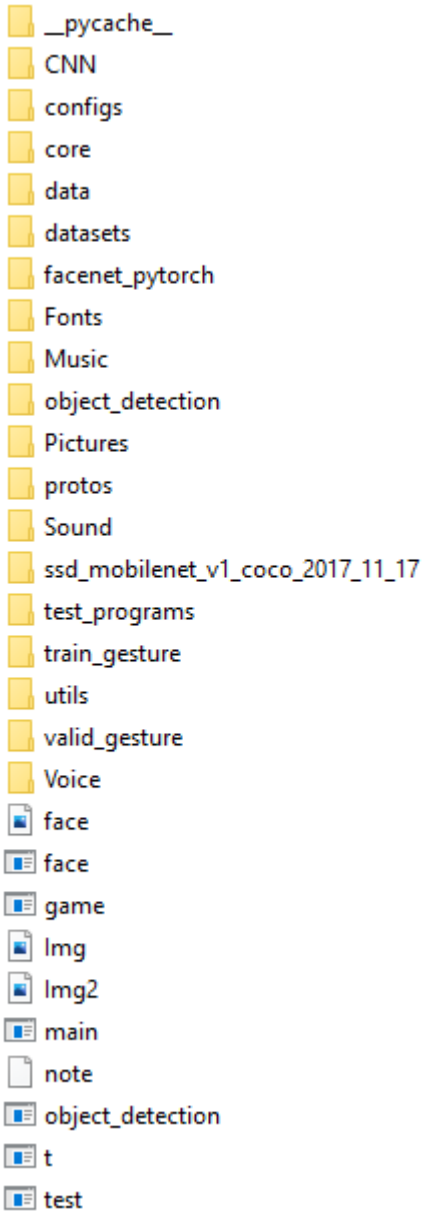
Integration and Testing

CNN – contains the models

test_programs – contains snippets of program

To run the whole program:

1) Open a Terminal
2) cd Desktop/FYP
3) workon py3cv4
4) python game.py



Key Technical Issues & Solutions

Blue/Orange tinted camera screen

The video frame has a blue tint after converting the from BGR2RGB, and orange tint if there is no frame conversion. This problem was only present after the switch to the Jetson Xavier NX. The main cause is not confirmed, however, the compatibility of OpenCV and the Jetson Xavier NX is suspected.

The problem was solved by adding a time delay after starting the camera stream, then setting the camera exposure level.

Low camera FPS

After the switch to the Jetson Xavier NX, the colour stream only has 2FPS. This has significantly reduced the accuracy of the object detection and facial expression recognition functions.

By forcefully setting the FPS in the program, we can counter this problem. However, the camera is only at 5FPS after the change. While it can be sufficient, changing to a better camera that can support at least 30FPS with the 2 functions will yield much better results.

Unresponsive buttons

The button is unresponsive, and often times, the user has to click the button many times for it to register. There are also times when there is a delay in the click registry, thus immediately skipping some screens, and going right into the game program.

By disabling the music in the program, the buttons are responsive, along with the entire program begin smoother to run.

Conclusion

Accomplishments

Gesture detection swapped from colour to depth stream, also streamlining the program, as well as increasing the accuracy and range of the gesture detection model

Implemented the object detection program for the eventual autonomous robot. Also works as a minigame at the end of the program.

Implemented the facial expression recognition program as a way to deliver a more interactable and engageable program.

Implemented several text-to-speech functions in the program to let the program seem more interactive and humanoid.

Bug fixes, optimization and improved program GUI,

Further Improvements

1) Currently, the program can only detect gestures in the cropped box. It is possible to instead zone in on the user's hand and scan the gesture. This way, the user will have more freedom as to how he wants to throw out a gesture (front-facing,side-facing,etc). However, this will also mean that the model has to be retrained with images of gesture from all about 90 degrees from the front. This will mean that the model will become exponentially bigger, and contribute to the already present slow loading times.
2) The concept of speech recognition is an important factor to deliver a more interactive program. Speech recognition will also remove the need to have buttons, which can reduce the memory usage of the program. To implement this, another model has to be introduced to the program. This will require a good isolating microphone.
3) Hardware/Software improvement. Since the upgrade to the Jetson Xavier NX, memory is no longer an issue. However, processing speeds are still relatively slow. An upgrade to the current Jetson device and the current Orbbec camera will be a big factor in ensuring the game runs smoothly, and at a high accuracy detection rate as well.

Appendix

Function Codes

Facial Expression Recognition

Object Detection

Augmenting

Depth camera + cropping

Text2Speech

Gaussian Thresholding

Game Program

Gesture Model Training

Running the Program

To run the main program:

1) Open a Terminal
2) cd Desktop/FYP
3) workon py3cv4
4) python game.py

To run a specific program:

1) Open a Terminal
2) cd ‘Directory’
3) workon py3cv4
4) python filename.py

References

https://github.com/priya-dwivedi/face_and_emotion_detection/blob/master/Facial%20Detection%2C%20Recognition%20and%20Emotion%20Detection.md - Facial Recognition Expression

https://pythonprogramming.net/tensorflow-object-detection-api-self-driving-car/ - Object Detection

https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html - Gaussian Threshold

https://readthedocs.org/projects/astra-wiki/downloads/pdf/latest/ - Depth Camera

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html - Image Augmentation, Model Training