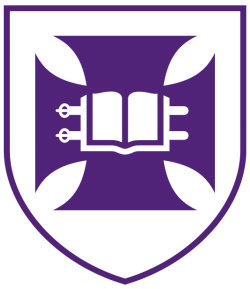
THE UNIVERSITY OF QUEENSLAND
FACULTY OF ENGINEERING, ARCHITECHTURE AND INFORMATION TECHNOLOGY



THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

DECO3801 PROJECT REPORT

Exploring a Universal Gestural Language

Brisbane city, 10/2023

Contents

1 Introduction

Our project aims to create a universal gestural language used to interact with multiple user interfaces. Using AI gestural detection technology, it will allow the user to control several commonplace scenarios on the computer such as media, video calls and general navigation using a gestural language that is based on affordance, pantomime and simple signs taken from Auslan.

The project will provide alternative input methods for people with mobility impairments, allowing them to interact with technology more effectively - helping to make computer systems more inclusive.

The project also offers hands-free interaction with the computer, useful for situations where users cannot or should not touch computer peripherals. This will allow users to perform tasks without physical contact, improving hygiene and convenience.

2 Methodology

2.1 Tools Used

2.1.1 Mediapipe

Mediapipe [2] is a open-source framework created by Google that provides various machine learning solutions with simple-to-use abstractions. Specifically, the project uses 'Hand Landmark Detection', 'Pose Landmark Detection' and 'Face Landmark Detection' (Fig **??**). These machine learning solutions detect specific joints and their connections on a given image or video.

(a) Hand Landmark Detection (b) Pose Landmark Detection (c) Face Landmark Detection

Figure 1: The Used Mediapipe Solutions

Mediapipe is a useful tool for the project as it is a easy to use framework which saves us from having to implement our own computer vision solutions which might have dubious results. Having a solution that detects landmarks is useful as training neural network models to detect gestures is much easier if the training data is a discrete number of coordinates opposed to training off images which runs into problems such as: lighting, getting a more diverse training set, varying camera dimensions, etc.

2.1.2 Tensorflow

Tensorflow [3] is an open-source python library which implements machine learning and artificial intelligence methods. It is used in the project to train of landmark coordinates to detect specific gestures using a neural network. Specifically a long short-term memory network

(LSTM). This is because we to detect the AUSLAN alphabet where many of the signs involve movement - a sequence of landmarks - which is best suited for a LSTM.

2.1.3 OpenCV

OpenCV (Open Computer Vision Library) is a open-source library that provides methods for real-time computer vision. In our case, openCV is used for image processing. It reads the input from the camera and feeds it into mediapipe to make detections. OpenCV also has great functions for drawing geometric shapes onto images which will be used to give visual feedback to the user.

2.1.4 Languages

The project uses Python for the back-end and JavaScript for the front-end. Python is used as it is a intuitive language that all team members are familiar with and has access to the aforementioned libraries. JavaScript is used for front-end as Python's GUI libraries are difficult to work with and we prefer JavaScript.

2.1.5 Flask

In order to use Python as back-end and JavaScript as front-end, the project uses Flask [4] to allow communication between the two.

2.2 Project Management

2.2.1 Communication

Discord is the teams preferred choice of communication. It facilitates calls for meetings and multiple text channels effective communication. All team members use Discord on a regular basis, so response time is shortened.

2.2.2 Task Management

Notion was used to manage assignment of tasks. It allows team members to assign and create tasks for other team members so that we can keep track of what needs to be done and if the work load is even for all team members.

2.2.3 Version Control

GitHub was used for version control. It's a cloud-based service which is suitable so the team can submit and view contributions fast.

3 Design Choices

The aim is to design a gestural language which will control multiple facets of the computer and allow for interaction with multiple user interfaces. The design choices were made to fit this criteria whilst also keeping user experience in mind.

3.1 Gestures

3.1.1 Gesture Modes

The gestural inputs are broken up into different modes which detect for different gestures. These modes are: cursor, media, zoom and AUSLAN. The following sections will break these modes down further.

3.1.2 AUSLAN Input

For the prototype, we found it would be best to only implement the AUSLAN alphabet. This is enough to show off the capabilities of the project prototype without having to implement unnecessary whilst being enough to communicate words using the gestural language. A program

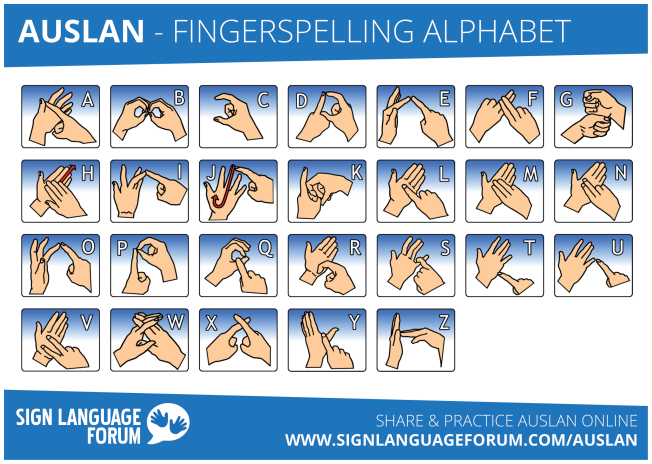


Figure 2: The Illustrated AUSLAN Alphabet

was created on the computer in order to capture 30 frames of data for 30 iterations, meaning there are a total of 900 data points. As such, there are 900 pieces of data per letter. The data being in "frame" blocks allows the program to be run in real time. This differs from the original vision of the prototype, in which it was going to be a "pause-select" system.

When first training the model, it was found that the model was extremely sensitive to data, likely occuring due to the large number of categories for the neural network to choose from. This was fixed by.....

3.1.3 Word Correction and Prediction

A depreciated component of the project. When inputting letters and words with the gestural language, it may be annoying to have to input the whole sentence and sometimes inaccuracy. To remedy this, we implemented a 'word prediction' and 'spelling checker' using machine learning

methods in tensorflow and GPT-2. But with time constraints and difficulty with training the whole AUSLAN alphabet, the word prediction component of the project had no use and was since depreciated.

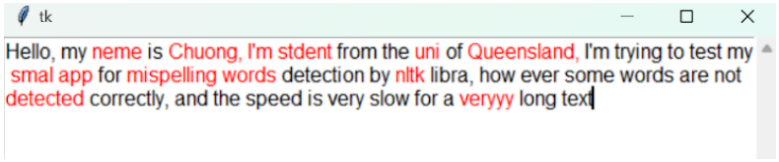
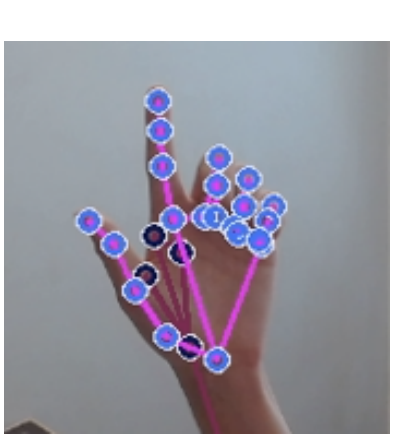


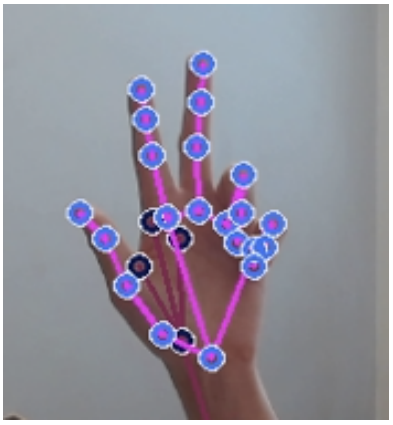
Figure 3: Spelling Checker

3.1.4 Mouse Input

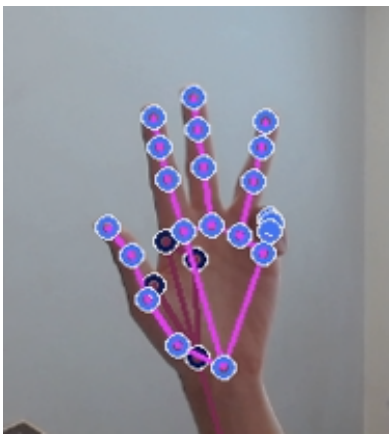
Mouse input was implemented as it allows for interaction with multiple user interfaces on the computer. The mouse gestures were designed to imitate the usual computer mouse that users are familiar to.



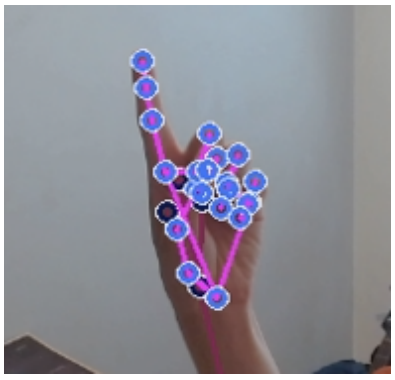
(a) Cursor: Left Click



(b) Cursor: Right Click



(c) Cursor: Scroll Wheel



(d) Activate Left Click

(e) Scrolling up

Figure 4: Cursor Gestures

To control the cursor, the user will have to either make the left click or right click gesture as indicated above. When the user moves their hand while making these signs, the cursor will move. To then activate the left click or right click, you simply cross your thumb over your palm.

Dragging is much like the computer mouse - activate the left click and hold for a little bit. Then you can move your hand to drag.

Scrolling is activated by raising the index, middle and ring finger and crossing your thumb. When holding this position, the user can move up or down to scroll up or down indicated by fig 4 (b). The further the hand is from the initial point (indicated by the white dot), the faster the scrolling.

3.1.5 Model

As aforementioned, the original plan for Gesture was to have the entire Auslan alphabet trained and ready for predictions.

3.1.5.a Media Control Gestures

When the user is consuming media such as video or audio, they can switch to 'media mode' which detects for gestures specific to control media. These include pause/play toggle, skipping forwards, skipping backwards and turning subtitles on.

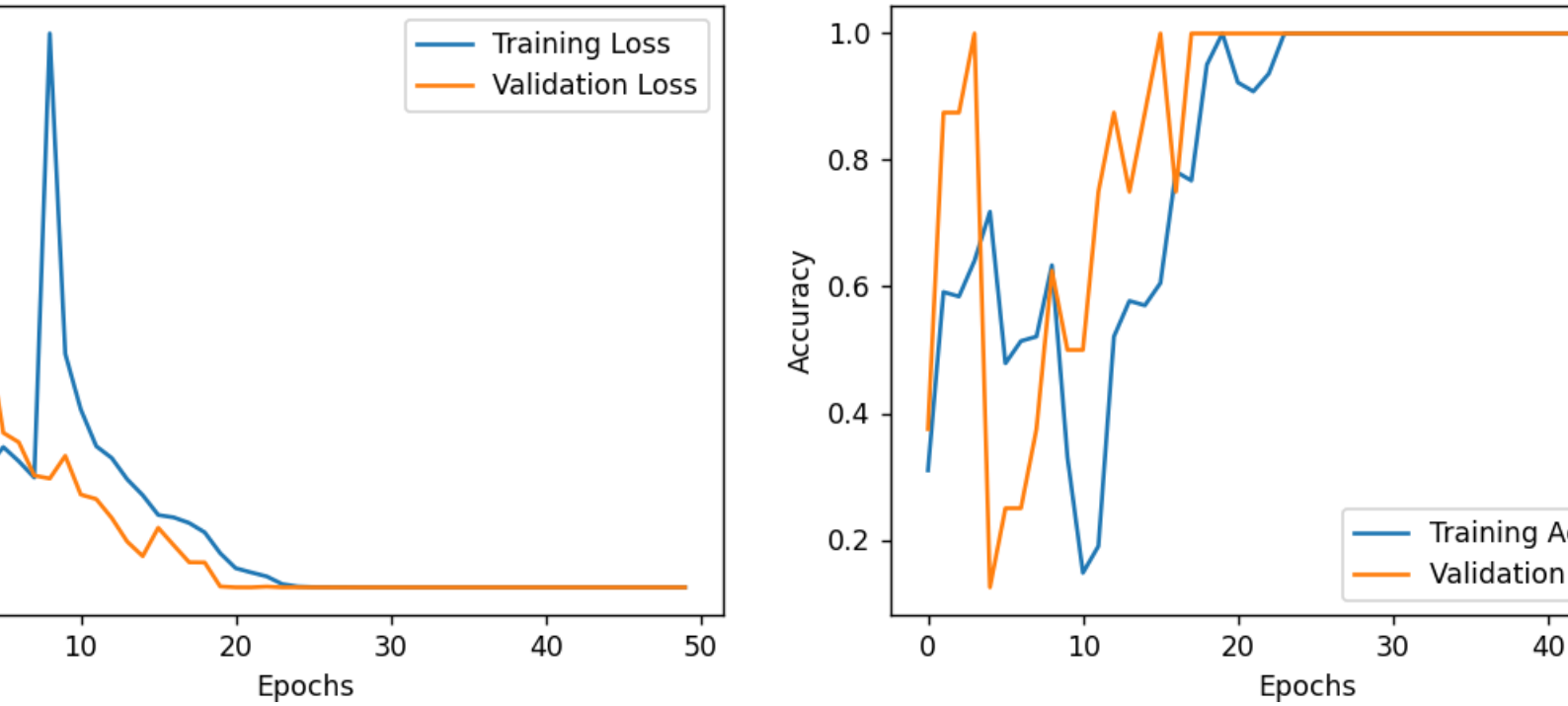


Figure 5: Labelled diagram of the bar, including interface at X_0

3.1.5.b Zoom Gestures

The zoom gesture mode is used for zoom video calls and will detect gestures that will toggle microphone, toggle video and end call.

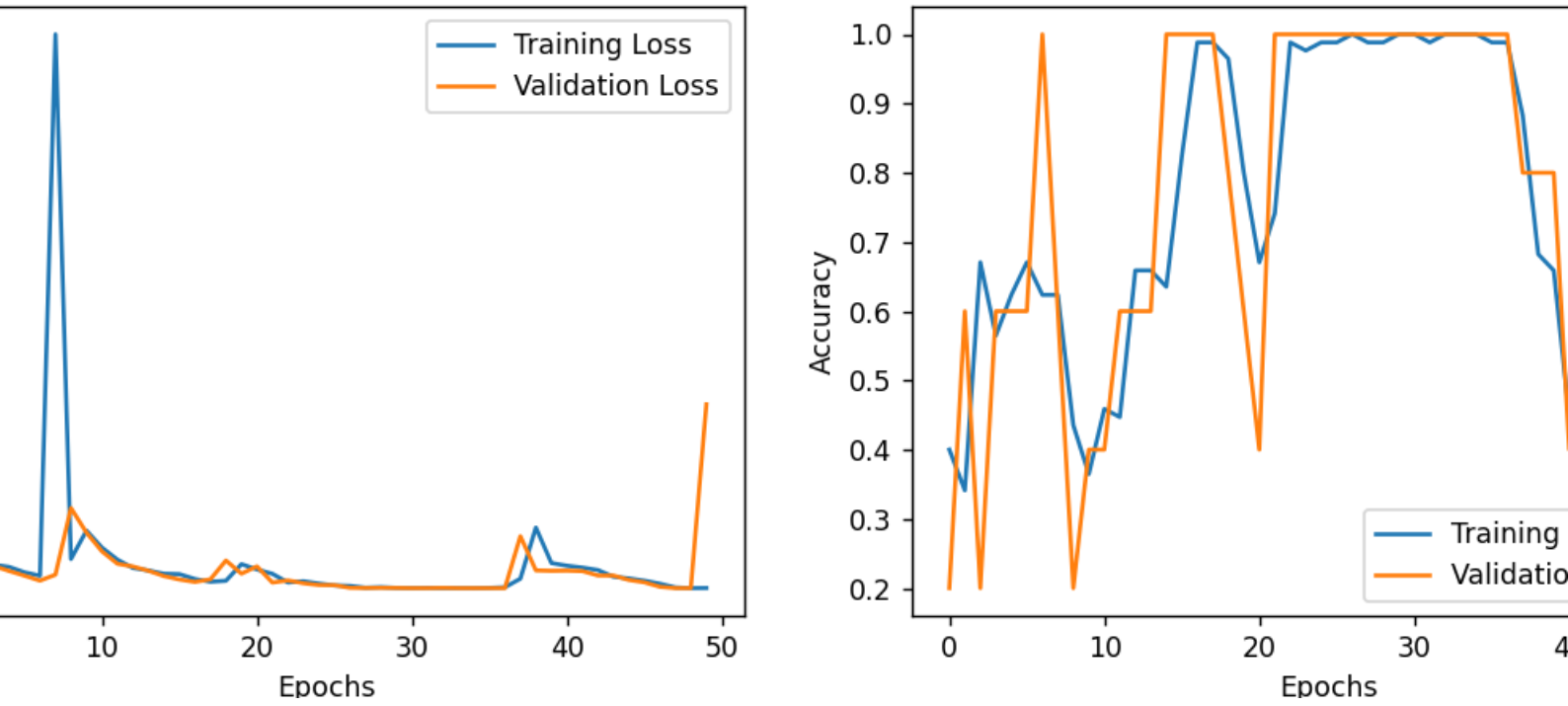
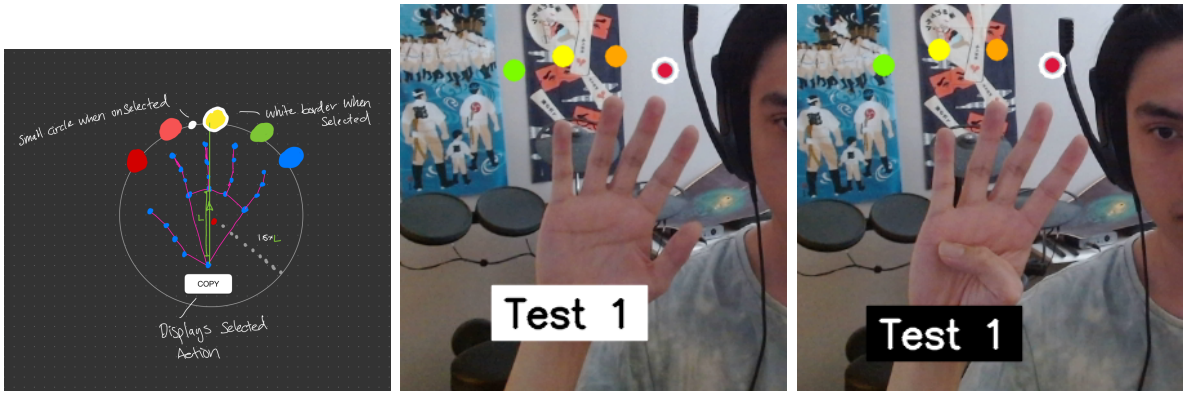


Figure 6: Labelled diagram of the bar, including interface at X_0

3.1.6 Quick Menu

The quick menu provides a way to activate multiple actions with a simple gesture. The quick menu is activated by raising your left hand with an open palm, showing buttons in a circle around the hand. This gesture somewhat imitates a dial which will be intuitive for users.

A cursor appears showing the current selection, which is moved by rotating the hand. When you are ready to make an action, simply cross your thumb over your palm and a selection will be made.



(a) Quick Menu Sketch (b) Quick Menu (c) Selection Made in QM

Figure 7: Quick Menu Demonstration

The quick menu was added as it creates an easy way to make actions with interesting visual feedback. It is also used to switch between the different 'gesture modes'.

Detection of raised fingers was originally detected using machine learning training but found that it wasn't fast enough to provide seamless feedback to the user. Instead, a method of using calculations on the landmarks was used and it is much faster.

The design choice was made to scale the circle with the distance from the camera as we want the user to interact with the computer as far as they want from the camera. Having a fixed menu radius creates a visibility problem - that the quick menu can go off the screen, so it was found that scaling the menu was better.

To avoid miss-inputs with the quick menu, a buffer-time was added between actions so that the user doesn't accidentally activate multiple actions at once. As well as this, the user must keep their thumb on their palm for at least 1 second before the action is selected.

3.1.7 On-Screen Keyboard

We thought that users might find inputting AUSLAN might get tiresome - especially when inputting single letters. So we made it so the user can bring up the windows on-screen keyboard. This is accessed using the quickmenu when in cursor mode.

3.2 UI/UX

3.3 Ethics, Security and Data Privacy

There are two main parts of the project that raise ethical, security and data privacy concerns. Our program requires the user must provide their camera input and allow to give control of their computer.

Although user is being recorded, they are able to view what the camera sees and their camera feed will only be used by the program on the computer - no data is being stored or used elsewhere.

In the current build of the project, users cannot give their consent, but in the future they should be made aware that their gestures are being recorded and are being used to control their camera.

Even though the project uses machine learning methods to train gestures, the project will not train off images retrieved from the user.

Another security concern is that the project takes control over the users computer system - which may be exploited by malicious actors. The program does not communicate with the internet but the project may still be modified to create viruses. In the future, this can be mitigated by encrypting the project.

The follow was the risk matrix that was created in the statement of work.

		IMPACT			
		Acceptable	**Tolerable**	**Unacceptable**	**Intolerable**
LIKELIHOOD	**Improbable**		Low accuracy on training.		-Data Leaks/Invasion of privacy - Lose access to github repo
	Possible	Team member unavailable	Poor runtime	Resource intensive	Incorrect/Losing Training Data
	Probable				- App is unrunnable

Figure 8: Risk Matrix

The only ethical, security and data privacy concerns we identified are data leaks and invasion of privacy. As discussed, we designed an offline project that doesn't store any information from the user, so there are minimal risk with regards to this.

However, if in the future we want to expand the project to be able connect to the internet and use user data, this would be a major concern. To limit risk, the user should be informed of how their data is used and proper encryption should be used.

4 Evaluation

4.1 Competitive Landscape

4.2 Stakeholders

5 Conclusion

6 References

References

[1] Code link:
https://github.com/lambdarm/deco3801

[2] Mediapipe:
https://github.com/google/mediapipe

[3] Tensorflow:
https://github.com/tensorflow/tensorflow

[4] Flask:
https://flask.palletsprojects.com/en/3.0.x/