DEVELOPMENT OF SIGN LANGUAGE DETECTION AND IDENTIFICATION BASED ON PYTHON

NUR IFFAH MAISARAH BINTI AZMAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF SIGN LANGUAGE DETECTION AND IDENTIFICATION BASED ON PYTHON

NUR IFFAH MAISARAH BINTI AZMAN

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek	:	Development of Sign Language Detection And Identification Based on Python
Sesi Pengajian	:	2024/2025

Saya NUR IFFAH MAISARAH BINTI AZMAN mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.
/	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS)

(COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: ....
....
....
....
....

PROF. MADYA TS. DR. NORHASHIMAH MOHD SAAD
Profesor Madya
Jabatan Teknologi Kejuruteraan Elektronik dan Komputer
Fakulti Teknologi dan Kejuruteraan Elektronik dan Komputer
Universiti Teknikal Malaysia Melaka

Tarikh : 24 January 2025

Tarikh : 24 January 2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled Development of Sign Language Detection And Identification Based on Python is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	
Student Name	:	NUR IFFAH MAISARAH BINTI AZMAN
Date	:	24 January 2025

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electronics Engineering Technology with Honours.

Signature	:	
Supervisor Name	:	ASSOC. PROF. TS. DR. NORHASHIMAH BINTI MOHD SAAD
Date	:	24 January 2025
Signature	:	
Co-Supervisor Name (if any)	:	
Date	:	

DEDICATION

This thesis is dedicated to my family, whose unwavering support, love, and encouragement have been my guiding light throughout my academic journey. To my parents, for believing in me and always pushing me to strive for excellence. To my friends, for their companionship and understanding, especially during challenging times.

Lastly, to my supervisor and mentors at Universiti Teknikal Malaysia Melaka, thank you for your invaluable guidance, encouragement, and insight, which have greatly shaped my growth as a student and as an individual.

ABSTRACT

Speech is the world's primary form of communication. However, deaf or hard-of-hearing people find it difficult to communicate because they have to use sign language. There is only one method available for their communication and such a common method of communication in gesture is sign language. Sign language is a form of communication used by hearing-impaired communities. This project aims to develop a system for hearing and deaf people to communicate more easily by developing a system that turns sign language into text. This project focuses on creating an automated sign language recognition system using Python to improve communication. The system combines hand landmark detection with a random forest classifier to recognize and interpret American Sign Language (ASL) gestures. Using a dataset of ASL gestures, the model was trained on 80% of the data and tested on 20%, achieving 97.01% accuracy. To solve this gap, this study develops to make sign language easier to communicate with other people.

ABSTRAK

Ucapan adalah bentuk komunikasi utama di dunia. Namun, golongan pekak atau kurang pendengaran menghadapi kesukaran untuk berkomunikasi kerana mereka perlu menggunakan bahasa isyarat. Hanya terdapat satu kaedah yang tersedia untuk komunikasi mereka dan kaedah komunikasi yang biasa dalam isyarat ialah bahasa isyarat. Bahasa isyarat adalah bentuk komunikasi yang digunakan oleh komuniti kurang pendengaran. Projek ini bertujuan untuk membangunkan sistem bagi orang yang mendengar dan pekak supaya dapat berkomunikasi dengan lebih mudah dengan membangunkan sistem yang menukarkan bahasa isyarat kepada teks. Projek ini memberi tumpuan kepada penciptaan sistem pengecaman bahasa isyarat automatik menggunakan Python untuk meningkatkan komunikasi. Sistem ini menggabungkan pengesanan tanda aras tangan dengan pengelas hutan rawak untuk mengenali dan mentafsir isyarat Bahasa Isyarat Amerika (ASL). Menggunakan set data isyarat ASL, model ini dilatih menggunakan 80% data dan diuji pada 20% data, mencapai ketepatan 97.01%. Untuk menyelesaikan jurang ini, kajian ini dibangunkan untuk menjadikan bahasa isyarat lebih mudah untuk berkomunikasi dengan orang lain.

ACKNOWLEDGEMENTS

In the name of Allah, the most Gracious and most Merciful.

Alhamdulillah, all praises to Him, for without His blessings and grace, this thesis would not have culminated in its completion.

I would like to express my deepest and most sincere gratitude to my supervisor, Assoc. Prof. Ts. Dr. Norhashimah Binti Mohd Saad, for her invaluable guidance, patience, and support throughout this project. Her encouragement and insights have been instrumental in the successful completion of this work.

My highest appreciation goes to my beloved parents, my Mommy and Daddy, as well as my family members, my brother and my little brother for their endless love, prayers, and unwavering support during the entire period of my studies.

An honourable mention also goes to my friends, whose motivation and understanding have provided me with strength and inspiration to overcome the challenges I faced along the way.

Lastly, I want to take a moment to thank myself for believing in my capabilities, for doing all the hard work, for never giving up, and for persevering through every obstacle. Alhamdulillah, I made it!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

%	- Percentage

LIST OF ABBREVIATIONS

TP	-	True Positive
TN	-	True Negative
FP	-	False Positive
FN	-	False Negative

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

According to the World Health Organization, approximately 430 million people have hearing loss and require some rehabilitation assistance [1]. Communication helps us to understand each other, solve problems and exchange more ideas. However, this privilege is not provided by those people who suffer from hearing-impaired and mute illnesses. To help disabled individuals communicate with deaf people more easily, current research has been expedited. A person who is hearing impaired is unable to hear as well as someone who has normal hearing thresholds in both ears, which are 20 dB or better. Hearing loss may be mild, moderate, severe, or profound. It can affect one ear or both ears and leads to difficulty in hearing conversational speech or loud sounds [1]. Deafness may occur from a variety of illnesses or injuries that affect speech or several hearing.

Sign language is the only way of communication using body movements especially hand sign gestures to express in parallel with the speaker's thought. In addition, an interpreter is also used when communication happens between normal people who speak but do not understand sign languages. It is not a convenient method as the interpreter would not be around the whole time to interpret for these people.

To propose the development of sign language detection and identification based on Python for hearing-impaired people. This project's aim is to solve this gap and provide a platform from which it is easy to communicate in sign language and also learn it. The system will detect real-time hand signs or gestures. Thus, providing a textual output in words by displaying it on the screen accurately of sign language is effective communication for the hearing-impaired community.

1.2 Societal/Global Issue for Development Sign Language Detection and Identification Based On Python.

Providing easy-to-use sign language is the primary form of communication for many deaf or hard of hearing people. By developing sign language detection and identification systems, it becomes easier to incorporate sign language into various digital platforms, such as educational resources. This might encourage and eliminate communication barriers, allowing individuals who use sign language to participate more actively in various aspects of life, including education, work life, and social interaction. Sign language detection and identification systems can help communication between sign language users and non-users. This technology can be integrated into real-time translation applications, making it easier for people to understand each other, regardless of their language abilities. It can also be used in educational settings to help teachers and students communicate more effectively. Sign language detection and identification systems can be implemented in public services, such as emergency services, healthcare facilities, and government agencies effectively, to improve the learning of the deaf and hard of hearing.

1.3 Problem Statement

Sign Language is the communication barrier between individuals who do not understand or use sign language. Deaf and mute people use hand signs to communicate, hence normal people face many problems in recognizing their language through signs. Hence there is a need for systems which is to recognize the different signs and convey the information to normal people.

Besides that, the lack of communication in sign language is more challenging for normal people. Especially in certain regions or specialized domains such as medical, legal, or technical fields. This can make it difficult for sign language users to communicate effectively in situations where an interpreter is required, such as medical appointments, legal proceedings, or professional meetings.

Finally, sign language has very limited accessibility, especially in public spaces, services and digital platforms. Most of the accessibility is not designed for individuals who use sign language. This can include a lack of captioning or sign language interpretation in public announcements, media content and emergencies.

1.4 Project Objective

The main aim of this project is to propose sign language detection and identification. Specifically, the objectives are as follows:

a) To develop an automated sign language recognition for deaf people in understanding the language gestures using Python.

b) To analyze the features used for American Sign Language using the fusion of hand landmark based on machine learning technique.

c) To verify the performance of the developed system by testing it on a dataset of sign language gestures.

1.5 Scope of Project

The scope of this project is as follows:

a) Development of a sign language system recognition using a Python for deaf and dumb people.

b) This system will provide of comprehensive dataset of sign language gestures.

c) Design a hand gesture detection and tracking using Python, Mediapipe and openCV.

d) The hand gesture will be detected in real time and the textual output will be displayed on the screen.

1.6 Outline of the project

Chapter 1: Introduction

This chapter introduces the project, detailing the goals and importance of developing the detection and recognition of sign language Python-based system. It outlines the problem statement, objective, scope of the project and the potential impact of the project on the deaf and hard-of-hearing community.

Chapter 2: Literature Review

This chapter reviews current research and technologies related to sign language detection and recognition. It discusses various methods, accuracy percentages, and datasets that have been used in previous studies. The literature review helps to identify the shortcomings of the current state of the art and opportunities for improvement.

Chapter 3: Methodology

This chapter describes the approach and techniques used in the development of a sign language detection and recognition system. It includes descriptions of the data collection process, preprocessing steps, model selection, and training procedures. The methodology provides a comprehensive guide to project implementation.

Chapter 4: Result and Discussion

This chapter includes the results of the study as well as the figures and illustrations that match into the experiments that are shown in the pictures.

Chapter 5: Conclusion and Recommendations

This chapter summarizes the main results of the project and discusses the overall success in achieving the objectives of the project. It provides recommendations for further work and possible improvements. In addition, the commercial potential of the developed system is explored and strategies to bring it to market are proposed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A literature review is a critical overview and assessment of the body of work (academic articles, books, conferences, etc) that has already been published on a specific topic or research question. In order to present a summary of the state of knowledge in a particular subject or field, it is necessary to read, evaluate and synthesize relevant materials. In this chapter, reviews definitions, past studies, and projects regarding the Development of Sign Language Detection and Identification Based on Python.

2.2 Understanding [Global/Current Issue] in the Literature

Understanding current global issues and the literature is crucial for developing sign language detection and identification systems based on Python for people with hearing impairment or deafness involves recognizing the difficulties of limited accessibility to deaf and hard-of-hearing education, the requirement for individualized approaches to managing

The literature highlights the key challenge in this domain is bridging the communication gap between the deaf and hearing communities, which can lead to social isolation and limited access to information and services for the deaf community.

One of the critical aspects of sign language detection and identification is the availability of large and diverse datasets. Researchers have been working on creating and curating datasets that capture sign language gestures from various perspectives, angles, and lighting conditions.

2.3 Sign Language

2.3.1 Definition of Sign Language

Sign language is a visual language which is communicated through gestures rather than spoken words. It communicates by using visible signs such as hands, eyes, actions, and facial expressions. Sign languages are full-fledged natural languages with their own grammar and lexicon. About 300 different sign languages are used in the world [2]. People who were deaf, whether completely or partially, had to struggle with challenges in society and were often treated as outcasts. However, sign language has become important in helping the deaf and hard-of-hearing community close those gaps so they may express themselves and effectively interact with others.

2.3.2 History of Sign Language

The first person to create an official sign language for the deaf is credited to Pedro Ponce de León, a 16th-century Spanish Benedictine monk. His choice to communicate in sign language was not entirely original. Using hand signals, Native Americans were able to facilitate trade with Europeans and communicate with other tribes. Benedictine monks had used these as a communication tool during their regular silence moment

.

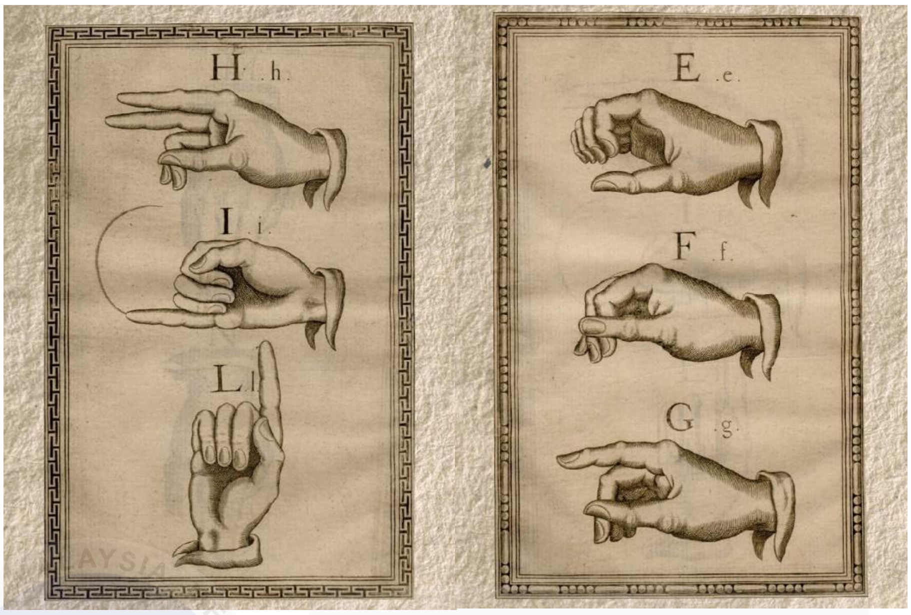


Figure 2.1 Illustrations of signs.

Ponce de León developed a method for teaching the deaf to communicate after being inspired by this latter practice to improve the gestures used in his monastery. This method has led to the development of systems that are today in use around the world. [3].

2.4 Gesture Recognition Sign Language

2.4.1 Gesture Recognition

Gesture recognition is a computer technique that recognizes and analyses human gestures using mathematical algorithms. Not only can gesture recognition be used to detect human hand movements, but it can also be used to recognize different walking movements and head nods.

In computer science, gesture and facial recognition is a fast-developing topic that is highlighted by an international conference. Gesture recognition technology can be applied to touch displays, cameras, and peripheral devices to enhance human-computer interaction.

Vision-based gesture recognition technology detects and translates user gestures in real time using a motion sensor and camera. Using gestures to track depth data is another feature of newer cameras and applications, which may improve gesture tracking. Real-time image processing helps users to interact with the program instantly and get the desired outcomes [4].

2.4.2 Hand Gesture Recognition

Hand gestures are highly useful forms of body language, the specific meaning of which is determined by the position and form of the fingers and palm through our language center. The gesture generally consists of two types of hand gestures such as static and dynamic. Dynamic hand gestures are produced through a pattern of hand movements, while static hand gestures are static shapes with a particular hand form. Depending on the individual and the specifics of their cultural upbringing, gestures and meanings may differ.

A hand gesture is a variety of movements and gestures produced with the hand, ha nd and arm, or both. It contains both static and dynamic hand gestures, the meanings of which are expressed through each gesture's form. Gesture and posture differ mainly in that gesture emphasises hand movement more than posture, which places greater focus on the form and condition of the hand and body [5].

2.5 Hand landmark

A hand landmark is a collection of key points on the hands that will be tracked or analyzed in different computer vision and machine learning applications. To aid with this, the landmarks are attached with particular areas of the hand including the wrist, finger tips and joints. This example uses the same model as the common landmark model used by Google's MediaPipe hands, which sources a model with 21 key points on the hand starting from the wrist to the tips of the fingers. Often the position of these landmarks is reported in a 3D space: (x, y, z) where x and y represent the horizontal and vertical position respectively and z represents depth or distance from the camera. Hand landmarks are an important tool in applications such as gesture recognition, sign language translation, and human-robot interaction since these features allow the systems to recognise and comprehend hand movements in real-time. [6]

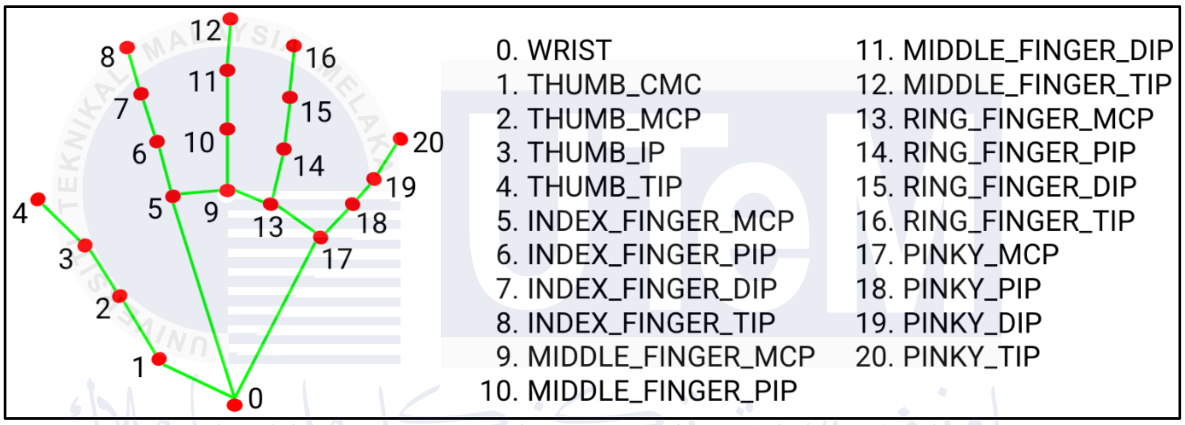


Figure 2.2 Hand Landmark

2.6 Scikit Learn

Scikit-learn is an open-source machine learning library for Python that offers random access matrix functionality, as well as datasets, algorithms, and simple, efficient tools that are as easy to use and comparable to the best commercial solutions for data analysis and modeling. There are many algorithms arranged based on classification, regression, clustering and dimensionality reduction. Scikit-learn is built ontop of other libraries like numpy, SciPy, and matplotlib which makes it easy to use any other data analysis library. As of its user-friendly interface and comprehensive documentation, it is widely used in academic research and industry applications. [7]

2.6.1 Random Forest classifier

A Random Forest Classifier is an ensemble learning algorithm used to perform classification tasks. During training, it builds multiple decision trees, and merge them in order to get an accurate and stable prediction. The main notion of the random forest is to enhance the generalization capability of the model by averting overfitting using an average of multiple decision trees. Random forests are robust to outliers and noisy data and give a good trade off between bias and variance and therefore perform well on a range of machine learning tasks. This is useful for large data sets, where a large number of features takes place. [8]

2.7 Literature Review based on Several Research Papers

The fundamental goal of a literature review is to connect the project idea with the knowledge and ideas that have been produced on a particular subject. Beginning in this investigation, a literature analysis was conducted to gather data on the technology available and the methodologies used by other studies that were already working on the same subject. There are multiple techniques to develop sign language.

2.7.1 Previous Research Paper

From the previous study, [9] proposed an article focused on Indian Sign Language (ISL) gestures are recognized using a real-time sign language recognition system. The segmentation function of OpenCV is used to identify and track the Regions of Interest (ROI) to identify signs. The hands' landmarks are obtained using Media Pipe, and a NumPy array is used to keep the landmarks' important points. In this project, TensorFlow, Keras, and LSTM are used to train the model. In addition, a live webcam stream can be used to test the model in real time. One potential application for the deaf and dumb is real-time sign detection and recognition, which enables them to interact with the outside world. In the past, the approaches for sign detection and recognition involved training machine learning algorithms on images. However, to improve real-time sign detection and recognition, deep learning models are currently being used. The final results proved that the suggested method had a 96% accuracy rate

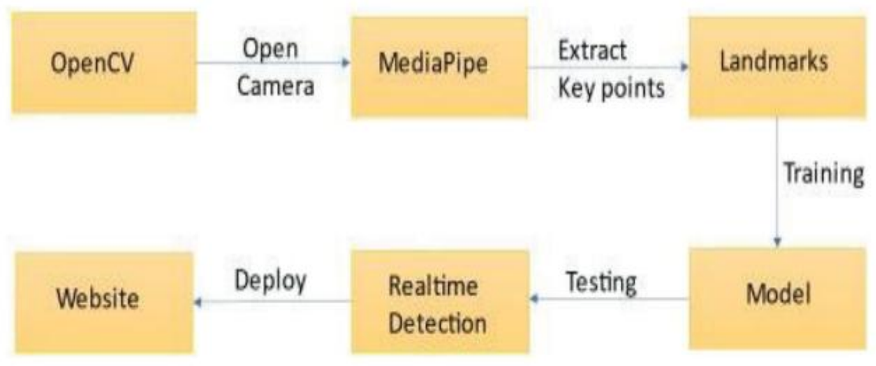


Figure 2.3 System Architecture

Another study, [10] proposed this project uses computer vision techniques for converting spoken or written language to sign language gestures and movements. By development of a Convolutional Neural Network (CNN)-based model that can identify and translate sign language gestures in real-time is in according to the fundamental ideas of Industry 5.0, where technology promotes variation and human contact while at the same improving processes.The accuracy of the system in this study is 91.67%, which is higher than the previous works in the literature.

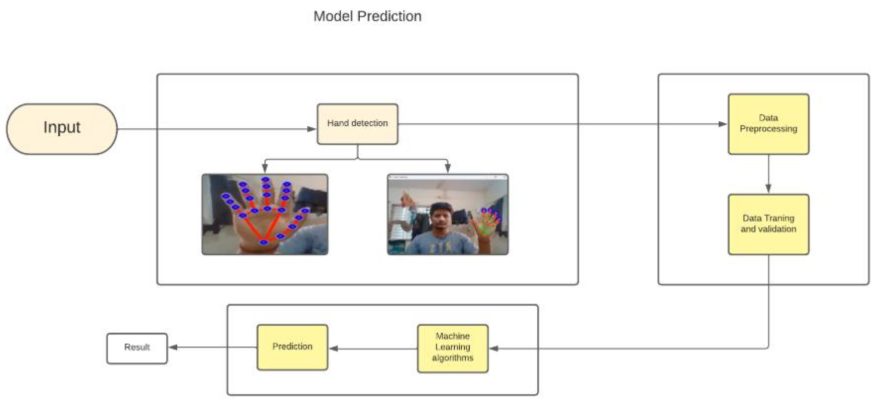


Figure 2.4 Proposed Architecture for Sign Language Recognition

In another previous research, [11] this paper introduced a method for detect American Sign Language (ASL) with the employing of Long Short-Term Memory (LSTM) networks, an approach called deep learning. The method suggested involves the use of a webcam and specifically developed software to record a dataset of ASL letters while allowing users to make ASL motions in front of the camera. An LSTM model was trained using the pre-processed photos, and it produced a 92% accuracy. This article focused on using the Mediapipe module to pre-process images and extract important features, and it proved to be a dependable and effective way to handle the amount of data.

From prior research, [12] suggests using a connectionist temporal classification (CTC) neural network-trained artificial neural network, specifically an RNN (Recurrent Neural Network) with LSTM (Recurrent Neural Network – Long Short-Term Memory), to translate speech into American Sign Language. In addition, it provides a gesture-based text/speech model which allows for two-way communication that can be set up without the use of an interpreter. This article uses the SSD Mobilenet V2 model for gesture recognition. The final results proved that the suggested method had a 84.46% accuracy rate.

Another research, [13] the project was to compare two feature extraction methods, SIFT (Scale Invariant Feature Transform) and SURF (Speeded Up Robust Features), for hand gesture recognition using depth maps acquired by a Microsoft Kinect camera. The goal of the study is to determine how effective these methods are in their ability to discriminate between hand gestures, which are essential to human computer interaction and gesture based control systems. A database of depth images was then created and the feature vectors were extracted, respectively, using SIFT and SURF in the experiments. A support vector machine (SVM) model for classification was trained using these vectors. The study results showed that SIFT had an accuracy of 81.2%, while SURF had the little higher accuracy of 82.8%.

Both methods had difficulties with gestures with similar shape or orientation, and their invariance to rotation was shown to be a disadvantage in some cases, the study concluded

From the previous research, [14] proposed that individuals who use Real-Time Sign Language Recognition (RTSLG) are able to talk in fewer phrases, express themselves more clearly, and depend on declarative language with greater expression. The article proposes a web browser-accessible deep learning model design that allows the use of Python, TensorFlow, OpenCV, and Histogram Equalization. The suggested RTSLG system identifies the features of the hand in webcam-filmed video by using image identification, computer vision, and neural network techniques, such as convolution neural networks. The suggested method achieves an accuracy of 87.8%. The suggested RTSLG system uses the gTTS (Google Text-to-Speech) library to change the displayed text to audio after the gesture has been identified and text output has been displayed. This will help those who are deaf or hard of hearing communicate.

In another research, [15] the project focuses on the design and implementation of a data glove system that utilizes Inertial Measurement Units (IMUs) to detect fine-grained hand shapes in real-time. The glove is equipped with multiple IMUs, a multiplexer, and a microprocessor, allowing for on-board computation and sensor data fusion to enhance accuracy and speed in gesture recognition. In this project, the accuracy of the gesture recognition system varied based on the number of participants used for training. The overall mean accuracy across all gestures was 92.4%.

Elsewhere, from the previous study, [16] proposed that system identifies a hand gesture as input and shows the detected character in real-time on the monitor screen. The goal of this project, which is classified as human-computer interaction (HCI), is to identify various alphabets (a-z), digits (0-9) and a number of common ISL hand gestures. This project was trained by Pre-Trained SSD Mobile Net V2 architecture using a custom dataset. In this project, various methods of posture recognition through computer interaction were examined and assessed. A combination of image processing techniques with human movement categorization was found to be the most effective method. The technology has an accuracy of 70-80% when detecting specific Sign Language signs in low light and without a controlled background.

In previous research, [17] proposed study introduces a new methodology for hand gesture identification based on Python scripts with media pipe, time, and cv2 (OpenCV) modules. Computer vision techniques such as hand gesture recognition enable computers to identify hand movements made by people. The technique suggested tracks and detects hand landmarks in real-time using the Mediapipe library. The final results proved that the suggested method had a 90% accuracy rate in identifying the number of fingers raised. This technique enables real-time gesture detection and hand tracking with just a common camera, with many possible applications in robotics, gaming, and healthcare.

Another research, [18] the aim of the project is to construct a Convolutional Neural Network (CNN) to be able to distinguish American Sign Language (ASL) fingerspelling from both color and depth images. To that end, the researchers used a deep learning architecture that accepts image intensity and depth data as separate inputs for enhanced feature extraction for classification. The ASL fingerspelling alphabet images were used to train and test the model excluding letters J and Z, as they convey meaning using motion. The CNN proved to be promising showing an average accuracy of 80.34%, a precision of 82% and a recall of 80%, which makes it useful in finger spelling recognition of ASL and brings us to take a step towards the automated interpretation of sign language.

According to the previous study, [19] proposed an automated system for recognition of sign language can help overcome this barrier by easily translating sign language movements into spoken English. A lot of people will find it easier to converse and communicate with the deaf and dumb. The technique we provide in this study uses a webcam to generate an Indian Sign Language dataset, which is then used to train a TensorFlow model via transfer learning to produce a real-time Sign Language Recognition system. The system has real-time sign language detection. Using Python and OpenCV, webcam images were collected for data acquisition, resulting in lower costs. An average accuracy rate of 85.45% is displayed by the developed system.

Elsewhere, from the previous study, [20] this paper proposed used United States Authorise Foreign Language to offer an Authorise Foreign Language acknowledgement within this specific study. The client must be able to take images of the property while it is moving using an internet-based camera for this particular study, and the gadget will predict and display the name of the photo that was taken. In order to classify the image, this study use of convolutional neural networks, or CNN. The model recommended is more precise and can recognise ten United States Authorised Motion Alphabets. In fact, the suggested version has achieved an amazing reliability of more than 90%.

In another research, [21] presents a deep learning model for Motion-Based Indian Sign Language (ISL) recognition. OpenCV is used to capture gestures, Media Pipe is used to identify the important points, and an LSTM (Long Short-Term Memory) model has been trained to predicts the sign. The proposed method can be related with video conferencing apps and used for real-time recognition of Indian Sign Language. A 92% accuracy rate for ISL recognition is a promising result for the proposed approach, and it opens up the opportunity for improvement in real-time situations using video processing.

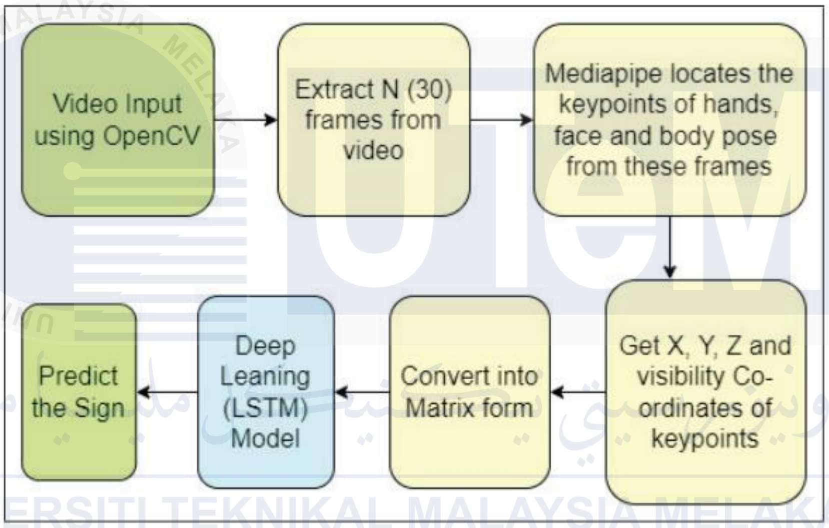

Video Input using OpenCV
Extract N (30) frames from video
Mediapipe locates the keypoints of hands, face and body pose from these frames
Get X, Y, Z and visibility Co-ordinates of keypoints
Convert into Matrix form
Deep Leaning (LSTM) Model
Predict the Sign

Figure 2.6 Flow Diagram

In another study,[22] proposed the developed a real-time hand sign language detection model that can recognize a wide variety of hand signs and Indian alphabet sign language, translating their meaning and providing us with an interpretation. In this paper, this model created using a Zoo-trained TensorFlow model, OpenCV, multiple Python libraries, and the TensorFlow object identification API. Using the camera and OpenCV, we have generated and use our own data set. To improve the model’s accuracy, we have produced a variety of data collections. For taking pictures, I used OpenCV with Python and a webcam. The system that was designed has an average efficiency of 85.4%.

From a preceding,[23] proposed that the main objective is to use a Human-Computer Interface (HCI) to identify human motions. A complex algorithm is required to translate these motions into computer language. Developing an automated system for human action recognition involves two main parts. The first step is To find out the frame sequences' characteristics. The second step is the action's classification. These representations will be used by a classifier to make distinctions between the various actions (or indicators). In this paper, this model uses   Convolutional Neural Networks (CNN), Python, OpenCV, Tensorflow, Keras and NumPY. A real-time hand gesture is used to create a human-computer interaction. With an average accuracy of 85.2%, it can effectively translate ASL to alphabetical letters in real-time.

2.8 The Comparison of Selected Literature Review

NO.	Article/Author	Year	Title	Method	Result	Summary
1.	A. Deep et al.	2022	Realtime Sign Language Detection and Recognition	• TensorFlow • Keras • Long Short-Term Memory (LSTM)	96% accuracy rate.	This article focused on an Indian Sign Language (ISL) gestures are recognized using a real-time sign language recognition system. The segmentation function of OpenCV is used to identify and track the Regions of Interest (ROI) to identify signs. The hands' landmarks are obtained using Media Pipe, and a NumPy array is used to keep the landmarks' important points. In this project, TensorFlow, Keras,

						and LSTM are used to train the model.
2.	H. Adhikari et al.	2023	A Sign Language Recognition System for Helping Disabled People	• Convolutional Neural Network (CNN)	Accuracy in sign language recognition 91.67% with real-time detection of sign language	This project converts spoken or written words into sign language motions and movements using computer vision algorithms. The creation of a Convolutional Neural Network (CNN)-based model that 22 recognizes and interprets sign language motions in real-time aligns with the fundamental concepts.
3.	R. Yeware et al.	2023	American Sign Language Detection Using Deep Learning	• Long Short-Term Memory (LSTM)	The pre-processed images were then used to train an LSTM	Method for detect American Sign Language (ASL) with the employing of Long Short-Term Memory (LSTM) networks, an approach called

					model, which achieved a remarkable 92% accuracy	deep learning. The method suggested involves the use of a webcam and specifically developed software to record a dataset of ASL letters while allowing users to make ASL motions in front of the camera
4.	C. Om Kumar et al.	2022	Real TimeDetection and Conversion of Gestures to Text and Speech to Sign System	• RNN (Recurrent Neural Network) • LSTM (Recurrent Neural Network - Long Short-Term Memory) • Connectionist Temporal Classification (CTC)	84.46% accuracy rate	In this paper, it suggests Using a connectionist temporal classification (CTC) neural network-trained artificial neural network, specifically an RNN (Recurrent Neural Network) with LSTM (Recurrent Neural Network - Long Short-Term Memory), to translate speech into American Sign Language.

5.	P. Sykora et al.	2014	Comparison of SIFT and SURF Methods for Use on Hand Gesture Recognition Based on Depth Map	• SIFT • SURF	82.8%	The aim was to test the ability to distinguish hand gestures for use in human-computer interaction or gesture-based control systems. A depth image database was constructed, and SIFT and SURF feature vectors were analyzed. To train a Support Vector Machine (SVM) classifier, these vectors were utilized.
6.	T. Siby et al.	2022	Gesture-based Real-Time Sign Language Recognition System	• TensorFlow • Histogram Equalization	87.8% accuracy rate.	The article proposes a web browser-accessible deep learning model design that allows the use of Python, TensorFlow, OpenCV, and Histogram Equalization. The suggested RTSLG system identifies the features of the

						hand in webcam-filmed video by using image identification, computer vision, and neural network techniques, such as convolution neural networks
7.	C. Mummadi et al.	2018	Real-Time and Embedded Detection of Hand Gestures with an IMU-Based Glove	• IMU-Based Glove	92.95%	The project involves designing a data glove system using Inertial Measurement Units (IMUs) to detect hand shapes in real-time. The glove includes IMUs, a multiplexer, and a microprocessor for on-board computation and accurate gesture recognition
8.	P. Verma, K. Badli	2022	Real-Time Sign Language Detection using TensorFlow, OpenCV and Python	• Pre-Trained SSD Mobile net V2	70-80% accuracy rate.	The goal of this project, which is classified as human-computer interaction (HCI), is to identify various alphabets (a-z), digits (0-9) and a number of common ISL hand gestures. This project was trained by

						Pre-Trained SSD Mobile Net V2 architecture using a custom dataset. In this project, various methods of posture recognition through -computer interaction were examined and assessed.
9.	A. Mukhopadhyay et al.	2023	Hand Gesture-Based Recognition System	• Media pipe	90% accuracy rate.	New methodology for hand gesture identification based on Python scripts with media pipe, time, and cv2 (OpenCV) modules. The technique suggested tracks and detects hand landmarks in real-time using the Media pipe library.
10.	S. Ameen et al.	2016	A Convolutional Neural Network to Classify American Sign Language Fingerspelling from Depth and Colour Images	• CNN	80.34%	The goal of the project was to create a Convolutional Neural Network (CNN) for recognizing American Sign Language (ASL) fingerspelling out of color and depth image. A deep learning

						architecture was used for better feature extraction where separate inputs were taken for image intensity and depth respectively. Training and testing were conducted on ASL alphabet images except J and Z that require motion.
11.	S. Srivastava et al.	2022	Sign Language Recognition System Using TensorFlow Object Detection API	• TensorFlow model via transfer learning.	85.45% accuracy rate.	The technique we provide in this study uses a webcam to generate an Indian Sign Language dataset, which is then used to train a TensorFlow model via transfer learning to produce a real-time Sign Language Recognition system. The system has real-time sign language detection. Using Python and OpenCV, webcam images were collected for data

						acquisition, resulting in lower costs
12.	S. Kandasamy et al.	2023	Sign Language Recognition using Python and OpenCV	• Convolutional neural networks, or CNN	90% accuracy rate.	This study use of convolutional neural networks, or CNN. The model recommended is more precise and can recognise ten United States Authorised Motion Alphabets.
13.	A. Ganpatye et al.	2022	Motion-Based Indian Sign Language Recognition Using Deep Learning	• Media Pipe • LSTM (Long Short-Term Memory)	92% accuracy rate.	OpenCV is used to capture the gesture, MediaPipe is used to identify the important points, and an LSTM (Long Short-Term Memory) model has been trained to predict the sign.
14.	N. Kumar et al.	2023	Hand Sign Detection Using Deep Learning Single Shot Detection Technique	• Zoo-trained TensorFlow model	85.4% accuracy rate.	In this paper, the model uses a Zoo-trained TensorFlow model, OpenCV, multiple Python libraries, and the

						TensorFlow object identification API. Using the camera and OpenCV, we have generated and use our own data set.
15.	Thiluckraj G K, Dr. M. N. Nachappa	2023	Sign Language Recognition By Image Processing	• Convolutional Neural Networks (CNN) • Keras • NumPY	85.2% accuracy rate.	These representations will be used by a classifier to make distinctions between the various actions (or indicators). In this paper, this model uses Convolutional Neural Networks (CNN), Python, OpenCV, Tensorflow, Keras and NumPY.

2.9 Summary

Finally, this project will benefit from a few papers comprising a literature review that explores existing research and studies relevant to the topic. It provides an overview of different approaches, methods, and technologies used in prior studies. The review discusses the technique and accuracy rate of sign language detection and identification. The review includes overall knowledge and progress in the development of this project. As a result, the use of sign language in these prior connected projects highlights the method and accuracy of each technique to develop sign language.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In general, this chapter will discuss research methodology and explore different frameworks, instruments techniques and methods. The methodology focuses on using Medi pipe and Python to create and build a machine learning model for sign language detection and identification. It follows step by step, starting with selecting a project title, conducting research, planning the project component and design, software, analyzing the project and writing a report. Understanding the software resources is important before starting the project.

3.2 Methodology

This is the aim of the project methodology "Development Sign Language Detection and Identification Based on Python" is to develop a system that consists of rephrasing sign language into text to ease communication between deaf and hearing individuals. The methodology aims to develop a user-friendly interface system that creates Python image detection. By incorporating these systems, the goal is to make it easier for sign language users and normal people. The primary objective is to eliminate communication barriers and improve education in sign language gestures.

3.3 System Block Diagram

Based on Figure 3.1, the block diagram shows that the system architecture for sign language detection and identification consists of several systems. At first, a sign language image dataset is obtained from Kaggle. The images are captured using a digital camera which creates a sign image. Then, the collected data is prepared for training by pre-processing data from input images using tasks such as landmark detection using tools like MediaPipe Hands. Once the data is preprocessed it is split into training and validation to be used in the data training and validation phase. For this step, train machine learning models such as Random Forest Classifier on this training and performance on the validation set. Lastly, testing the recognition and sign language recognition system and evaluating the results is a key step in the development process.

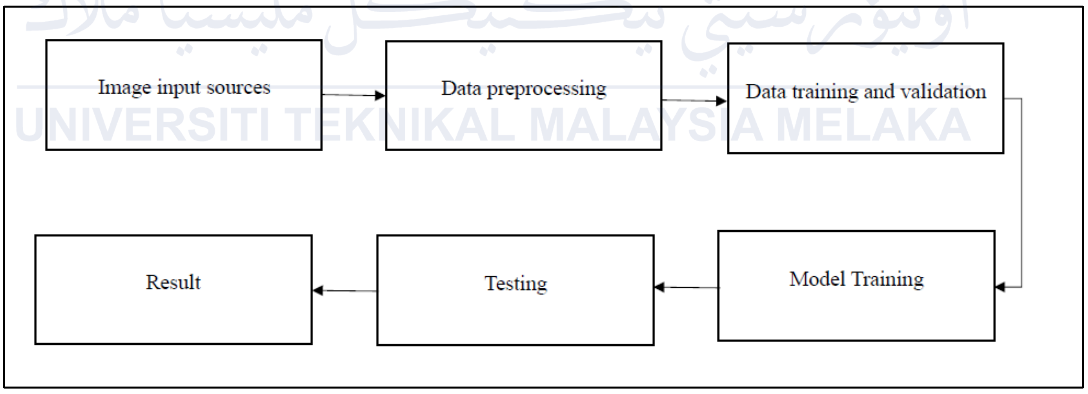


Figure 3.1 Block Diagram

3.4 Flowchart

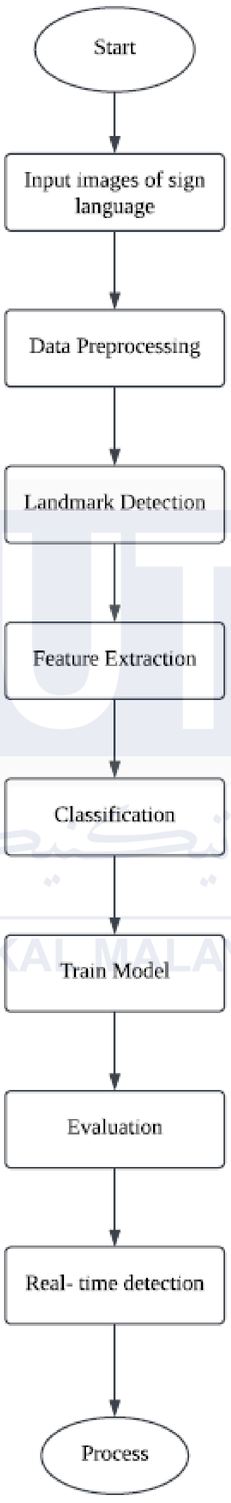
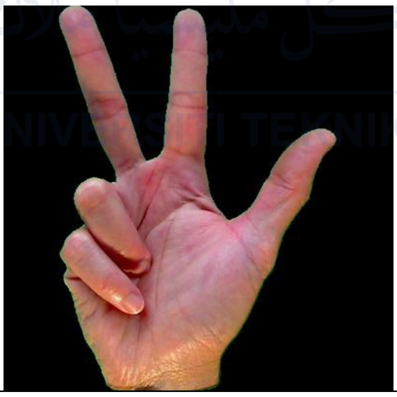
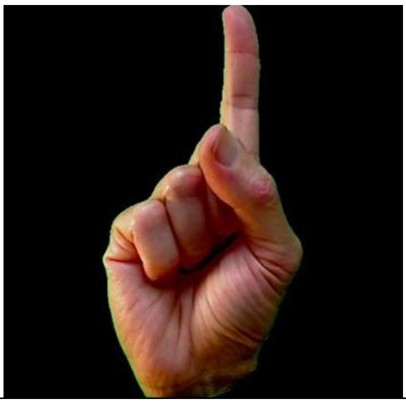


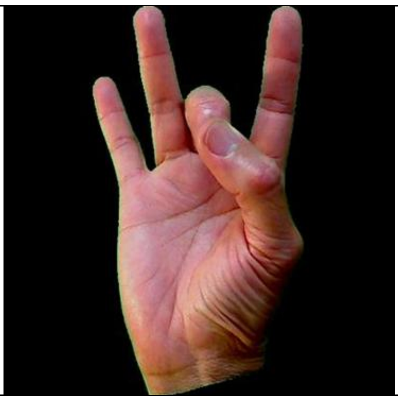
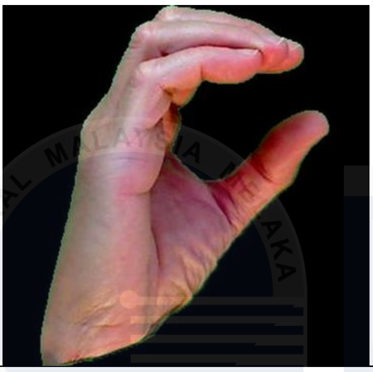
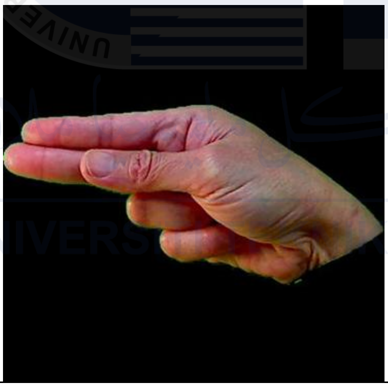
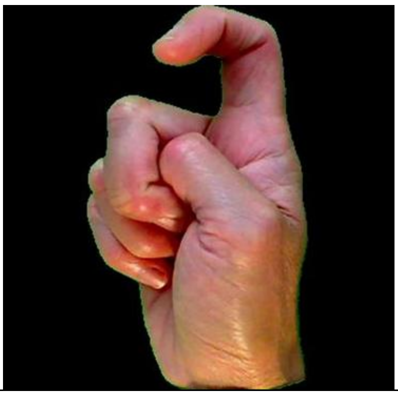
Figure 3.2 Flowchart of sign language detection

3.5 Dataset

This project's primary source is a compiled dataset of American Sign Language, or simply ASL, known as the ASL Alphabet created by Kaggle user Ayush Thakur [24]. The dataset consists of 2,520 photos which are, each of size 400x400 pixels. It has a total of 36 classes, each classes have 70 images, 10 for numbers and 26 for the alphabets A to Z. The hand gesture identification in the Media Pipe hands module is used as a media pipe handling the process of coordinates extraction and hand image normalization. The labels and data were then saved in a pickle file for use in training machine learning models. Table 3.1 depicts some sample photos of the classes in the dataset.

Table 3.1 Images of Dataset

Image	Classes Dataset label
	3
	1

	8
	C
	H
	X

3.6 Preprocessing

Data preprocessing refers to one particular data preparation operation, any raw data processing that is performed in advance of another data processing operation. As with the previous decades, it has been historically vital as the first step in data mining. More recently, data preparation methods were modified to train AI and machine learning models, and interpret their results from.

Preprocessing data is the process of altering data to make it better and more simply processed for data mining, machine learning, and other data scientific applications. To get the findings correct, most of these methods are applied at the very beginning of the machine learning and artificial intelligence development pipeline. For this project, the preprocessing refers to taking hand gestures data images and reshape them so that feature, or landmark, can be extracted to train a machine learning model. [25]

3.7 Feature Extract

Feature extraction is the process of reducing the dimensionality of raw data while keeping the essential features required for a given task. It involves finding and separating the most important or useful information from the data. In computer vision and machine learning, feature extraction converts raw input (such as text, audio signals, or images) into structured representations (such as patterns or numerical vectors) that algorithms can more readily understand and analyse. Efficient feature extraction maximises the value of the extracted data for tasks like classification, detection, or prediction while reducing redundant or irrelevant information. [26]

3.8 Detection

In general, the term detection is meant generally to describe the determination of the existence of objects, features, or patterns existing in data. This representation can be an image, movies, or even signals. The term "detection" describes a number of subtasks occurring in computer vision and signal processing that involve discovering if there exists an object or specific feature of interest and where it is in input. [27]

3.9 Identification

The identification is the process by which one determines or recognizes the specific category, class or identity of an object, signal or entity from its qualities or characteristics. In computer vision and machine learning, identification consists of giving the label or output to an input data depending on the pattern found in the trained model. [28]

.

3.10 Accuracy

Accuracy is one of the measures for a classification model. It involves calculating the proportion of correctly predicted cases, including true positives and true negatives—among all forecasts in relation to the actual output. in other words, this demonstrates the accuracy with which the model recognizes or categories the incoming data. [29] The formula for precision is given by:

$$\text{Accuracy} = \frac{Correct\ Prediction}{Total\ Prediction}$$

3.11 Evaluation Metric

In machine learning, evaluation metrics are numerical measurements used to measure how effective and how well a model is performing. In comparing different models or algorithms, these metrics provide very important information about how good a model is at predicting outcomes.The evaluation metric selection is dependent on the particular problem domain, data type, and desired result. [30]

3.11.1 Confusion Matrix

Confusion Matrix is a machine learning tool that is very important for measuring the performance of classification models. Its essence it is a table that displays the results of the predictions that a model returned against the actual outcome. The matrix is an easy to interpret visualization of how many instances were correctly, and incorrectly, classified by the model. For each row of a matrix, the instances in those rows belong to an actual class, and for each column, the instances in those columns belong to a predicted class. It is designed to make spotting the 'confusions' between classes obvious.

Assessment based upon True Positives (TP) or cases where the model correctly predicts the positive class, True Negatives (TN) Cases where it correctly predicts the negative class, False Positives (FP)Cases where it incorrectly predicts a positive class, and False Negatives (FN) Cases where it incorrectly predicts a negative class. These in turn allow us to calculate any number of performance metrics like accuracy, precision, recall, or F1 score that provide deeper insights about how effective a model is, rather than just the accuracy rates. [31]

3.11.2 Precision

Precision is measuring how accurate positive predictions are. It's defined as the ratio of true positive predictions to the sum of positive predictions (true positives + false positives). A high precision means a model has a low false positive rate. [32] The formula for precision is given by:

$$Precision = \frac{TP}{TP + FP}$$

3.11.3 Recall

Recall, also can be referred to as sensitivity or true positive rate, measures a model's ability to find all the relevant instances.It is a ratio of true positive predictions and the total number of actual positive (true positive + false negative). High recall means that a model is being bad at predicting false negative. [32] The formula for recall is given by:

$$Recall = \frac{TP}{TP + FN}$$

3.11.4 F1 Score

The F1 Score is a harmonic mean of both precision and recall, a single metric which comes with a balance between both concerns. However, it's very helpful when working with unbalanced datasets. The value is the F1 score, which ranges from 0, with perfect precision and recall, to 1. [33] The formula for F1 score is given by:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.12 Software Implementation

3.12.1 Python Language

Python is well-known programming language that is classified as high-level, general-purpose language, which is sections among web development, data analysis, artificial intelligence, automation and scientific computations. Python is easy to use and can be easily coded to test other algorithms in minutes; it is well suited for people who are not computer engineers. Python's simple syntax and readability make it simple to swiftly test complex algorithms, and it's also suitable for nonprogrammers [34].

Figure 3.3 Python Language

3.12.2 OpenCV

OpenCV or open Source computer vision is an open source computer vision and machine learning library. It enables developers to build applications which can work on images and videos, and give analysis on the same in real-time. The collection currently contains over 2500 optimised algorithms, including numerous basic and sophisticated approaches to computer vision and machine learning. These algorithms can be applied for searching similar images from a database of images, to recognize faces and objects and to classify behavioral patterns in the videos.[35].

Figure 3.4 OpenCV

3.12.3 Mediapipe

Media Pipe is an open-source solution that was developed by Google to create and run computational machine learning pipelines with special emphasis on real-time multimedia input including audio, video and photo streams. It allows the development of superior computer vision applications by developers because its architecture is segmented into calculators, which serves as building blocks. Some of these combined calculators can be strung together to produce a graph that shows flow, which sends processing tasks through efficiently for facial purposes, hand tracking, and object identification. [36]

3.13 Summary

This chapter presents the proposed methodology in order to develop sign language detection and identification. To ensure the success of the project implementation process, a project progress flow chart and block diagram was prepared. Before starting this project, the software was selected carefully. The further development of the project is effective and it is important to choose the software for created the machine learning method. As conclusion, the key part of this stage is the analysis and identification of all the software that is related to this project.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the results and analysis of the development of the detection and identification of sign language for deaf people. The system analyses and categorises sign language gestures using computer vision and machine learning algorithms. The goal of this is to solve this gap and provide a platform from which it is easy to communicate in sign language and also learn it. This part discusses the software step by step to produce the output by PyCharm. The result of the project can be seen by displaying it on the screen accurately in sign language.

4.2 Dataset Overview and Preprocessing

Dataset of 36 hand gestures (0-9, A-Z) were using the data from kaggle and Mediapipe was used to extract 21 hand landmarks for each of them. It contained 2,520 samples, each of which corresponds to 70 samples per gesture. It was split into 80% training and 20% testing. Table 4.1 shows the distribution of samples:

Table 4.1 The Distribution of Samples

Gesture	Samples
0	70
1	70
2	70
3	70
4	70
5	70
6	70
7	70
8	70
9	70
A	70
B	70
C	70
D	70
E	70
F	70

G	70
H	70
I	70
J	70
K	70
L	70
M	70
N	70
O	70
P	70
Q	70
R	70
S	70
T	70
U	70
V	70
W	70
X	70
Y	70
Z	70

4.2.1 Preprocessing

During the preprocessing stage of the sign language recognition system, the images are converted to RGB format and via Mediapipe are processed detecting hand landmarks. These landmarks have their coordinates normalized to relative positions in the image which allow to handle variations in hand size and distance from the camera. Only samples with exactly 84 values are kept to ensure data consistency (21 landmarks with 2 coordinates each), any remaining invalid data is discarded. Next, in the part of the code where video is captured in real time, due to the fact the hand poses are diverse helps the model to generalize better. Visualizing the data, like plotting the landmarks on images, can be used to assess the consistency and quality of the data.

4.3 Model Training

Media pipes are used to detect and use key hand landmarks for training models which have 21 points on each hand representing important hand points. These landmarks are converted to relative coordinates (x,y) and used as model features. In order to achieve the goal of this project , Random Forest Classifier (RFC) is selected for this project as it helps handle complex data and does not overfit by using multiple decision trees. Here, class_weight='balanced' weights underrepresented classes heavier during training. The more general test for the model’s performance on different subsets of the data is cross validation. The cross-validation result reveals that the proposed model works. Among other classifiers, RFC is chosen because it is simple, efficient, and performed better without additional fine tuning.

4.4 Result Analysis

4.4.1 Performance Metrics

Results indicate that model analyses perform very well with an overall accuracy of 97%. Predictions are strong and consistent, and most classes have high precision, recall, and F1–scores. Only a few classes like “F”, “G”, “0” show somewhat lower performance with F1 scores of 0.67, 0.86 and 0.86 respectively. The model seems to struggle a bit with these classes. Finally, the results indicate balanced and robust performance on all data variants, even as class sizes vary, with a macro average F1 score of 0.96 and a weighted average F1 score of 0.98. Overall performance is strong but results for minority classes show some issue with dealing with imbalanced classes. Table 4.2 shows that the Precision, recall, and F1 score for the testing set.

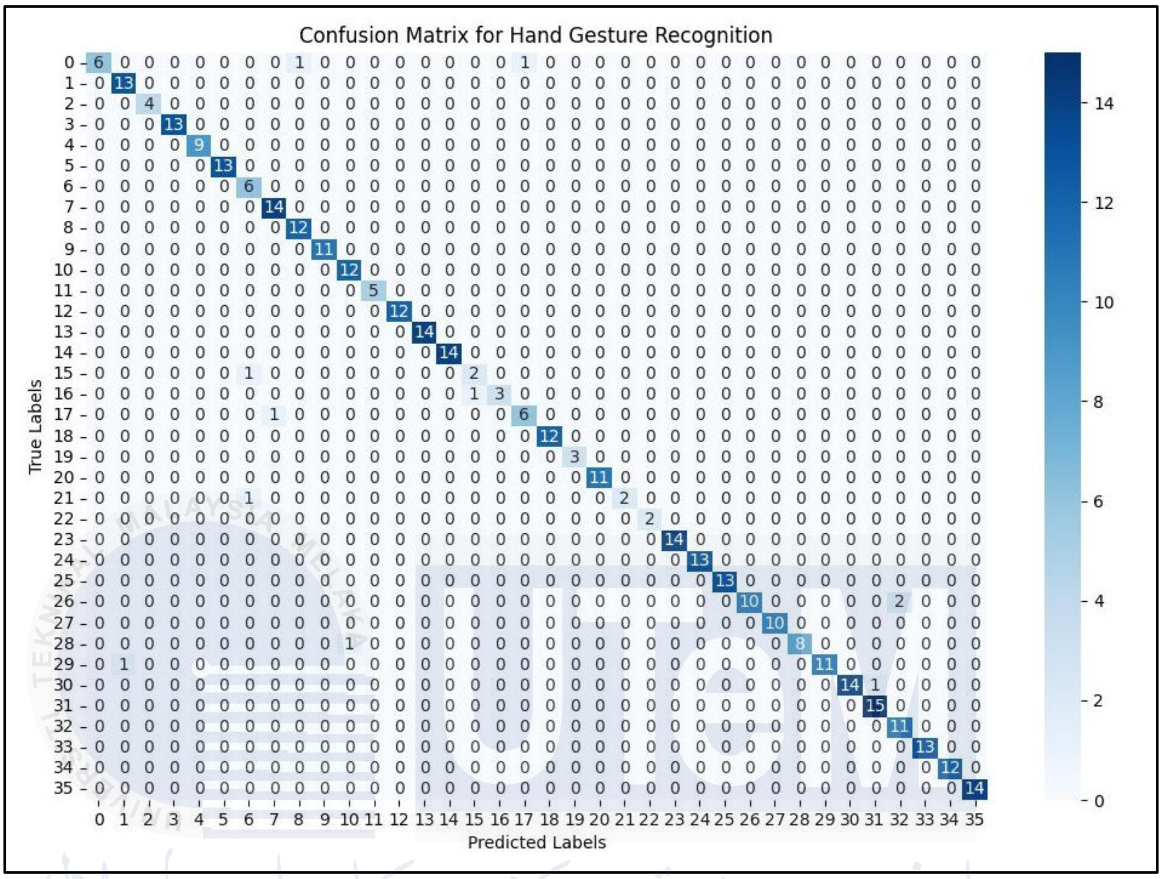
Table 4.2 Precision, recall, and F1 score for the testing set.

Class	Precision	Recall	F1-score	Support
0	0.75	1.00	0.86	8
1	1.00	0.93	0.96	13
2	1.00	1.00	1.00	4
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	9
5	1.00	1.00	1.00	13
6	1.00	0.75	0.86	6
7	1.00	0.93	0.96	14
8	1.00	0.92	0.96	12
9	1.00	1.00	1.00	11
A	1.00	0.92	1.00	12
B	1.00	1.00	1.00	5
C	1.00	1.00	1.00	12
D	1.00	1.00	1.00	14
E	1.00	1.00	1.00	14
F	0.67	0.67	0.67	3
G	0.75	1.00	0.86	4
H	0.86	0.86	0.86	7
I	1.00	1.00	1.00	12
J	1.00	1.00	1.00	3
K	1.00	1.00	1.00	11
L	0.67	1.00	0.80	3
M	1.00	1.00	1.00	2
N	1.00	1.00	1.00	14
O	1.00	1.00	1.00	13
P	1.00	1.00	1.00	13
Q	0.83	1.00	0.91	12
R	1.00	1.00	1.00	10
S	0.89	1.00	0.94	9
T	0.92	1.00	0.96	12
U	0.93	1.00	0.96	15
V	1.00	0.94	0.97	15
W	1.00	0.85	0.92	11
X	1.00	1.00	1.00	13
Y	1.00	1.00	1.00	12
Z	1.00	1.00	1.00	14
Accuracy			0.97	368
Macro avg	0.95	0.97	0.96	368
Weighted avg	0.98	0.99	0.98	368

4.4.2 Confusion Matrix

Confusion matrix for a hand gesture recognition based on Figure 4.1, confusion matrix for a hand gesture recognition system on a multi class classification model. The rows in this matrix correspond to our true labels (actual gestures), and the columns correspond to our predicted labels (model predictions). It shows the number of times a true label was predicted to a specific label in the matrix, one cell per each.

Most of the non zero values of this confusion matrix lie along the diagonal (where the true label matches the predicted label) which means the predictions are correct. This implies that the model would work efficiently for most gestures. The color in the diagonal cells is the darker the more correct, and the number of correct predictions is indicated with colors of increasing intensity. The misclassifications are represented by off diagonal cells which are minimal, implying low error rate.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	2	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
29	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	1	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14

Figure 4.1 Confusion Matrix for dataset

4.4.3 Comparison with Existing Methods

The table compares different methods used for sign language recognition based on features, datasets, and accuracy. Methods from previous years include using SURF and SIFT features in 2014, achieving 82.8% accuracy, and a CNN-based approach in 2016 with 80.34% accuracy using American Sign Language datasets. In 2018, an IMU-based glove system utilizing Inertial Measurement Units (IMUs) with French Sign Language (LSF) achieved 92.95% accuracy. The proposed method in this study uses a Random Forest Classifier with an American Sign Language dataset obtained from Kaggle, achieving the highest accuracy of 97.01%. This demonstrates that the proposed method outperforms previous approaches in terms of recognition accuracy.

Table 4.3 Result of Sign Language Detection and Identification

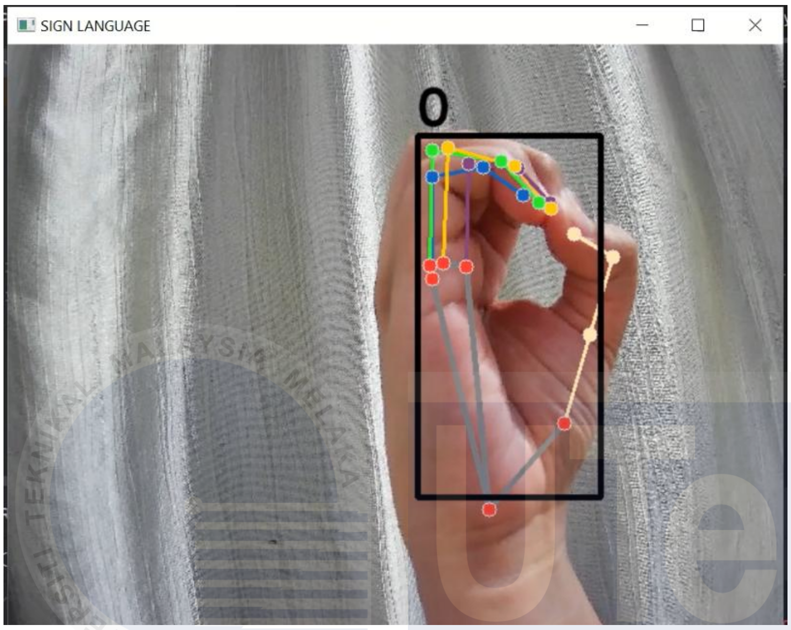
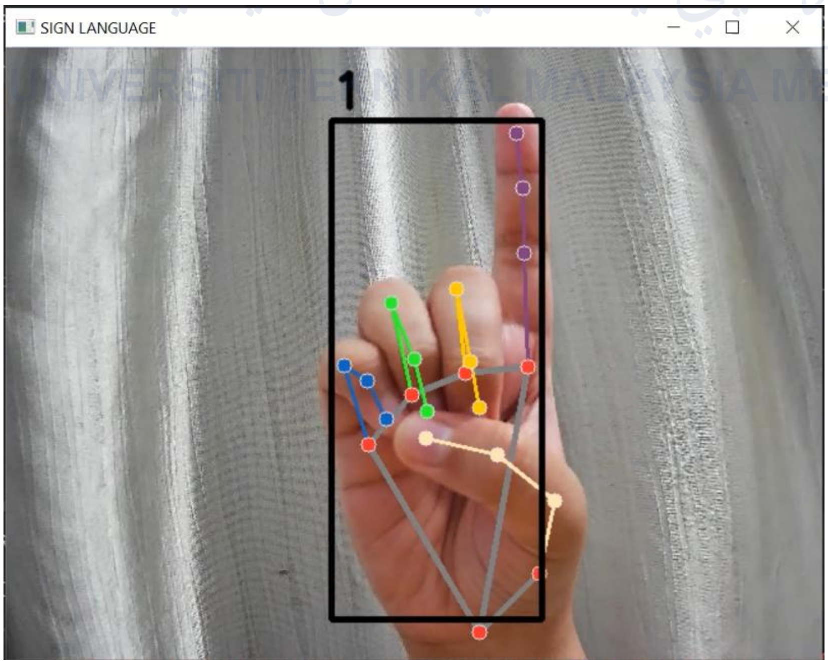
Year	Features	Dataset	Accuracy in (%)
2014	SURF and SIFT		82.8
2016	CNN	American sign languages	80.34
2018	IMU-based glove	Inertial Measurement Units (IMUs), French Sign Language (LSF)	92.95
Proposed Method	Random Forest Classier	American sign languages, Kaggle	97.01

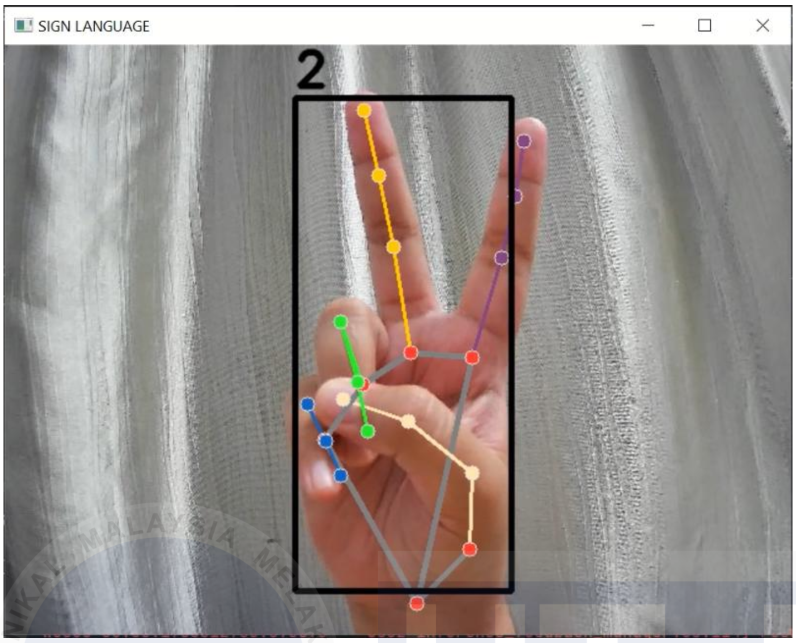
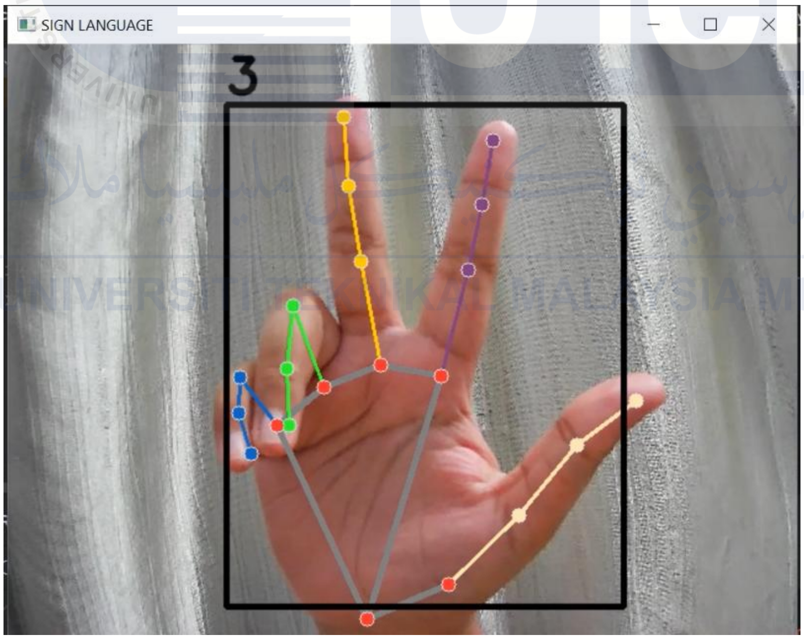
4.5 Result

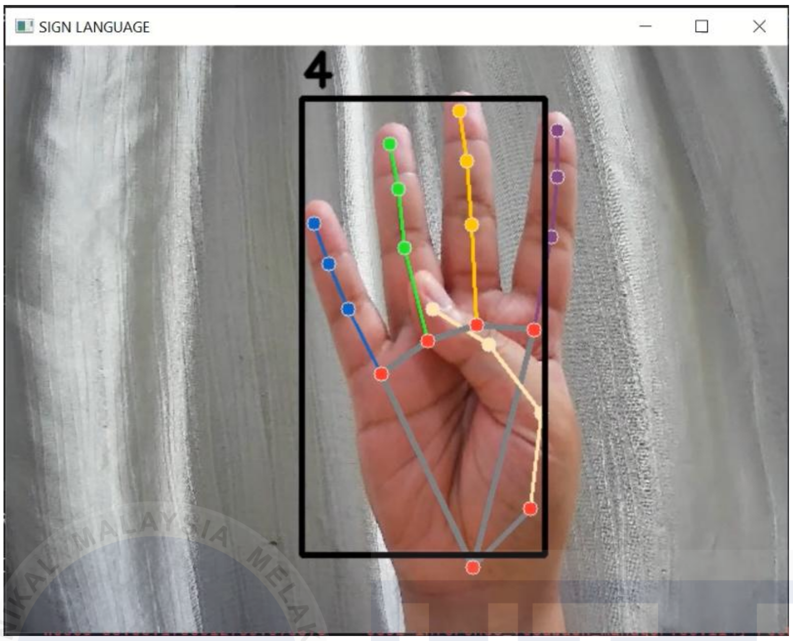
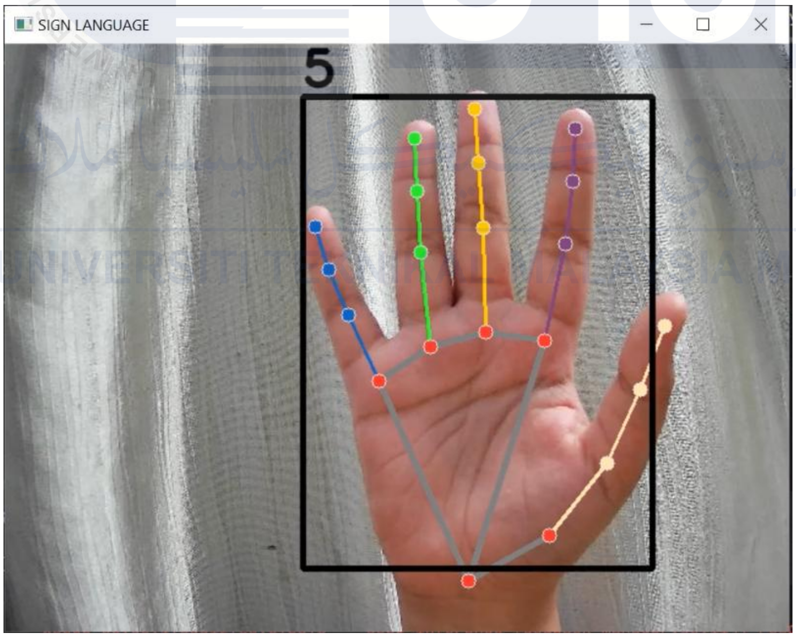
Based on Table 4.4, sign language detection and identification using Python, as demonstrated in the provided image. The system successfully detects and localizes the hand using a bounding box, indicating its ability to isolate the region of interest for further analysis. It identifies keypoints on the hand, such as finger joints and the palm base, and connects these points to form a skeletal structure. This visual representation helps map the spatial configuration of the hand. In the image, the system recognizes the hand gesture as the letter alphabet and number in sign language, as shown by the label within the bounding box. However, because of their same configurations, some hand motions are tricky to differentiate. For example, hand gestures of ‘D’ and the number ‘1’, ‘V’ and ‘2’, ‘W’ and ‘6’, and the letter ‘O’ and number ‘0’ are similar and confusing to the classifier. In contrast, letters with positions such as fingers on 'E', 'S' and 'T' are harder to be recognized accurately as these positions pose higher challenges for detection.

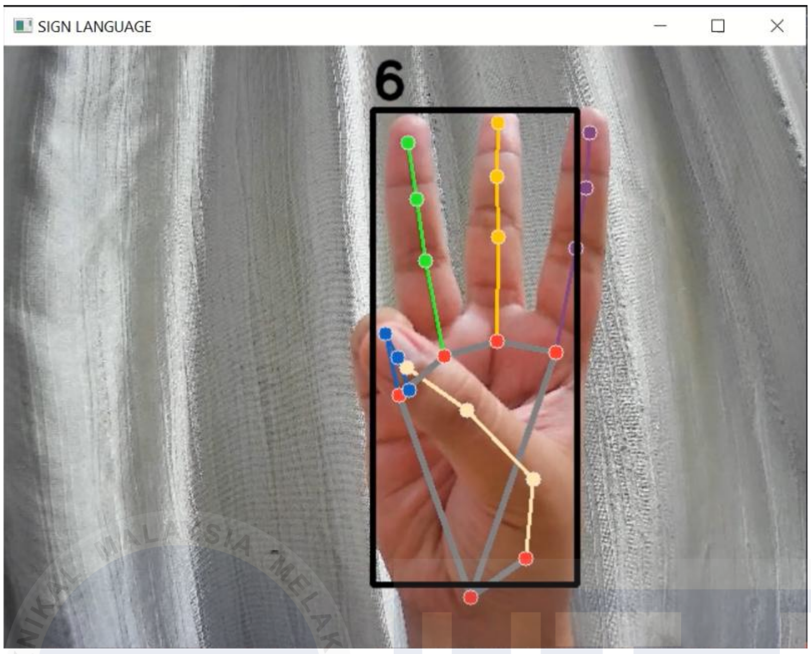
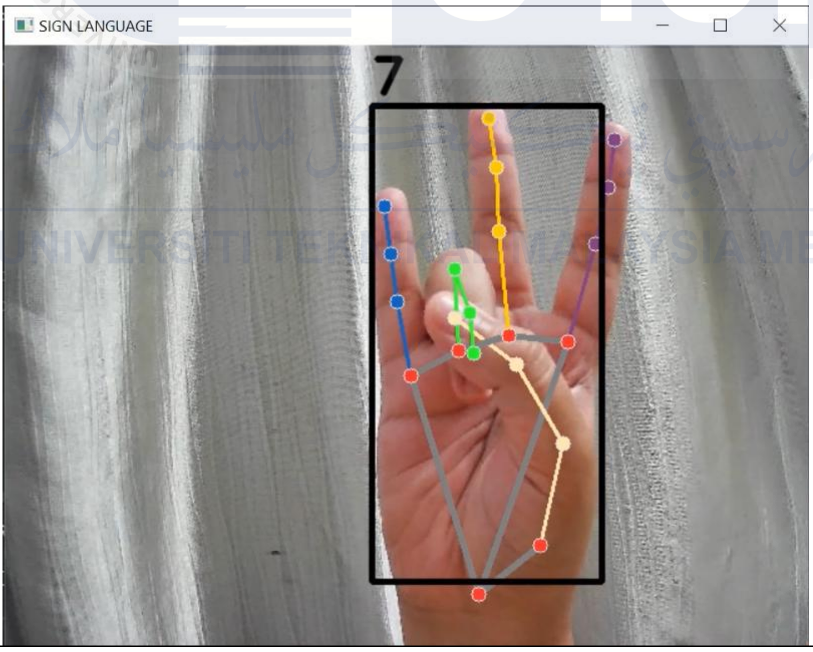
This recognition likely uses machine learning or deep learning models trained on sign language gestures, extracting features like distances between keypoints or angles of finger joints to classify the gesture. The system demonstrates robustness by effectively detecting the gesture against a textured background, suggesting strong preprocessing and detection capabilities. This project is a promising step toward enabling effective communication for individuals using sign language and can be expanded to include additional signs or real-time applications.

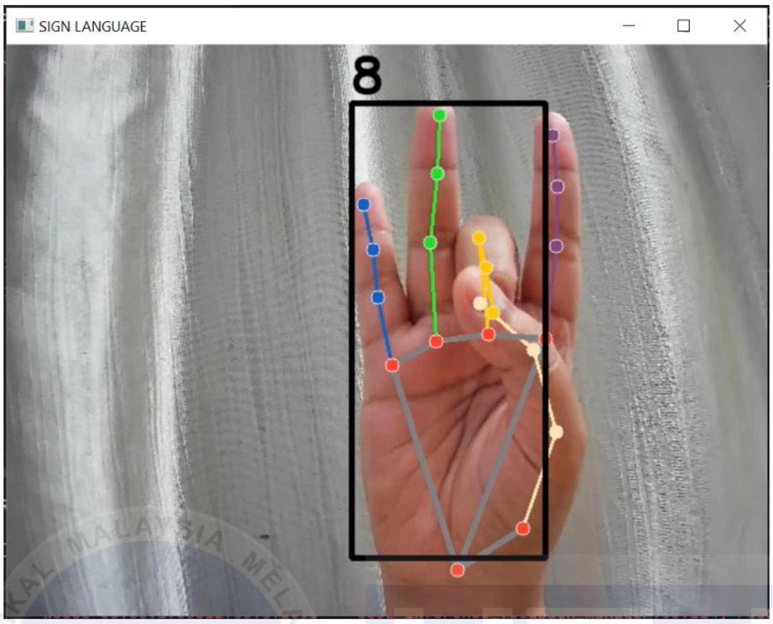
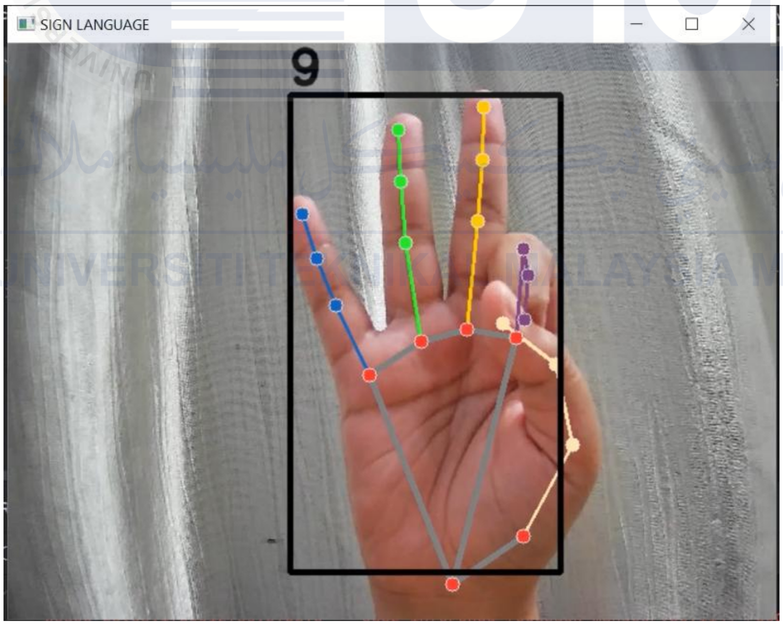
Table 4.4 Result of Sign Language Detection and Identification

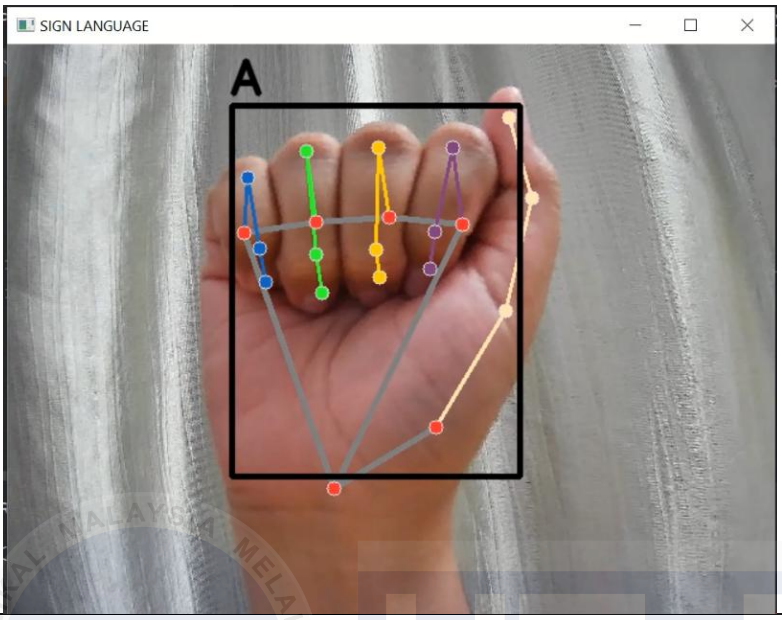
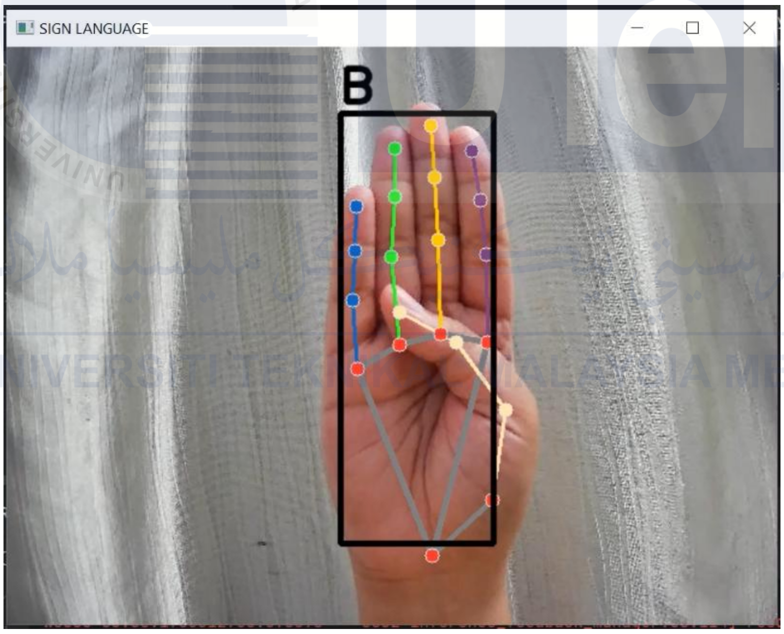
Result	Text Output
 SIGN LANGUAGE	0
 SIGN LANGUAGE	1

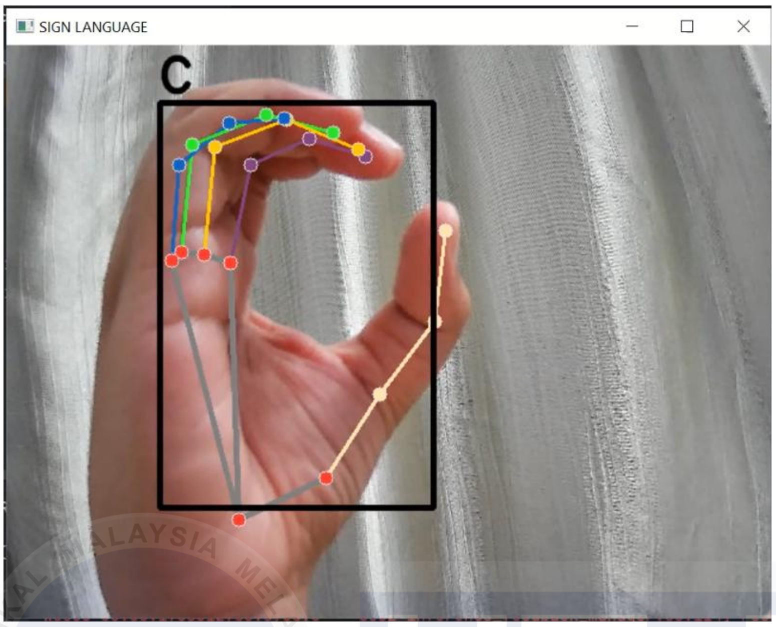
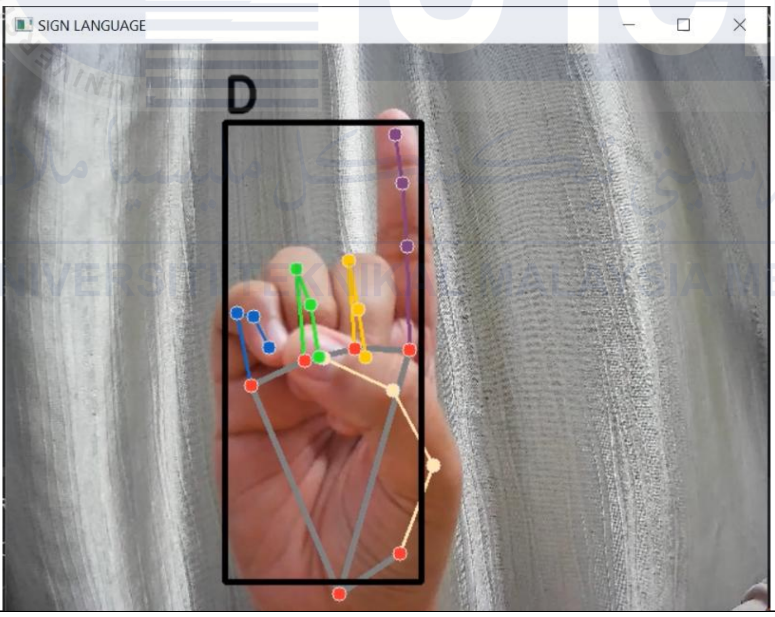
	2
	3

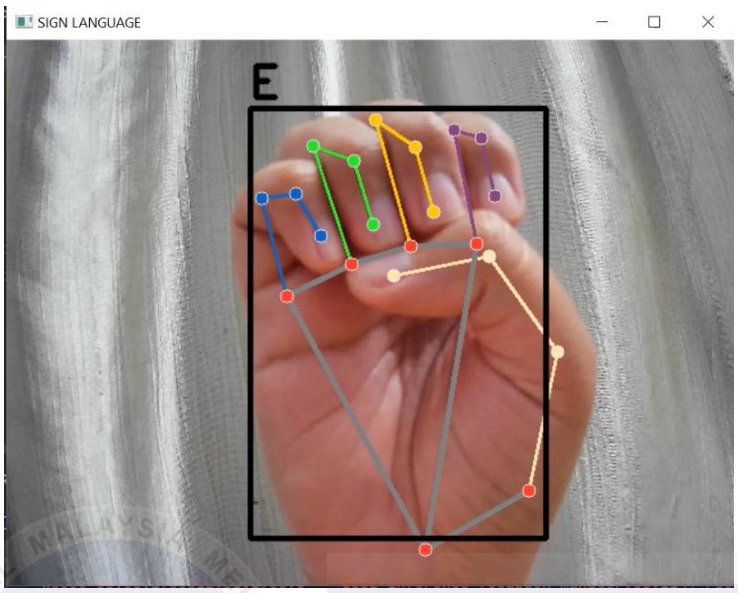
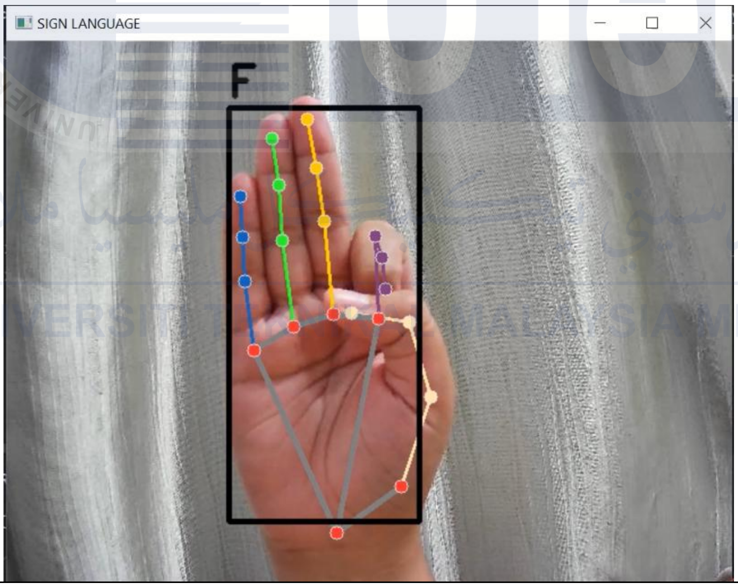
	4
	5

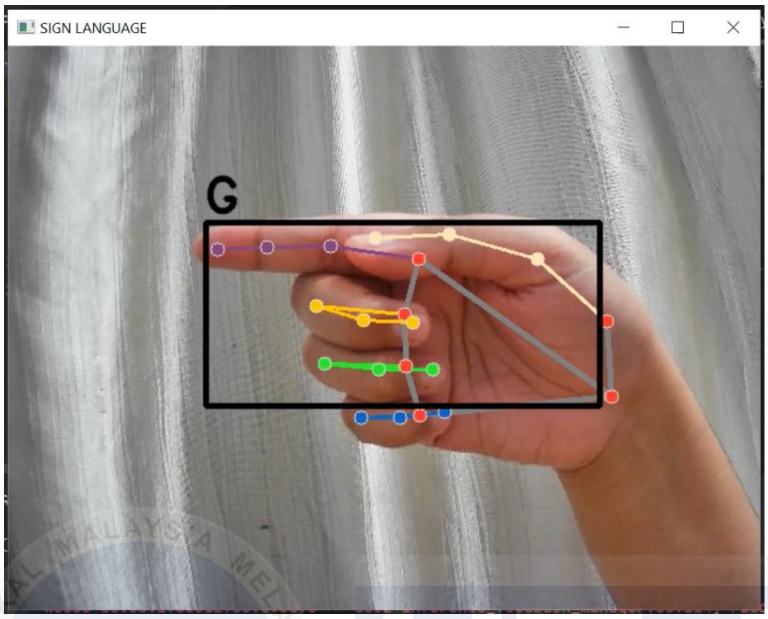
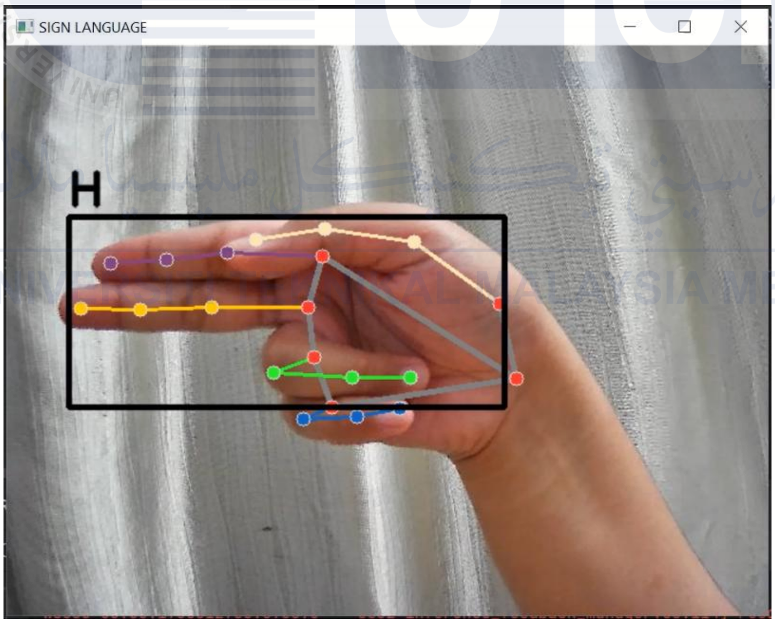
	6
	7

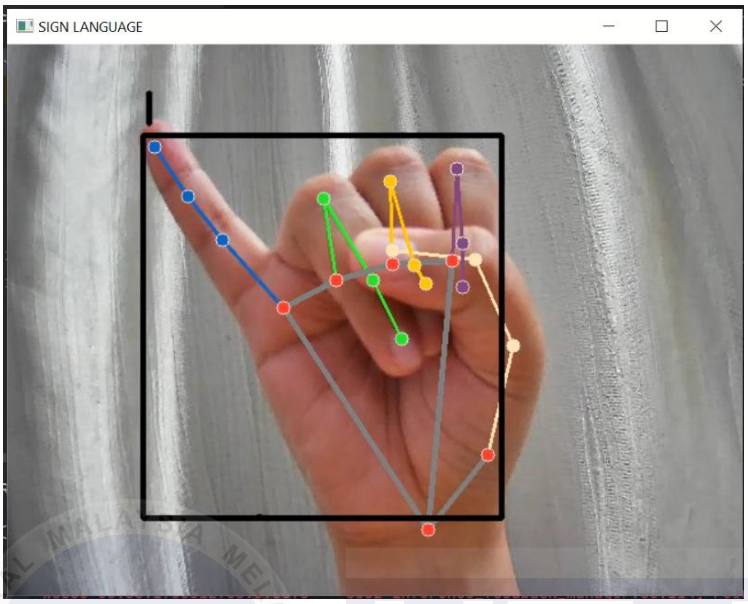
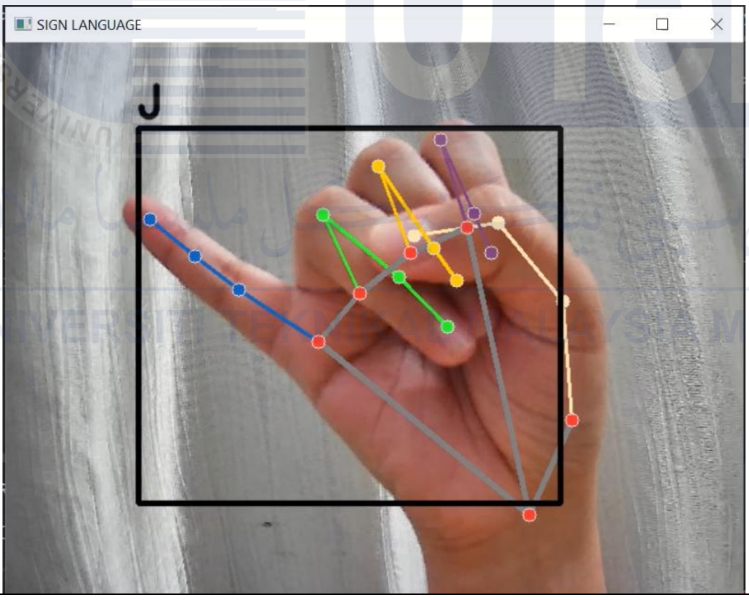
	8
	9

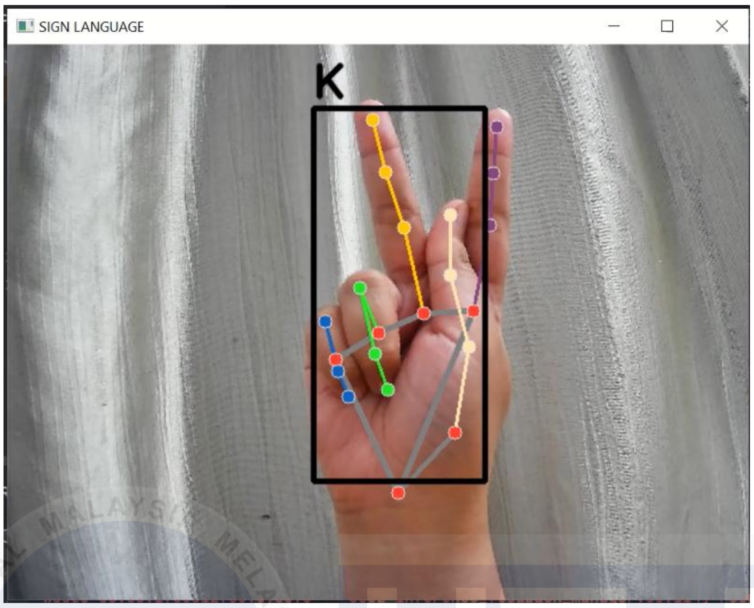
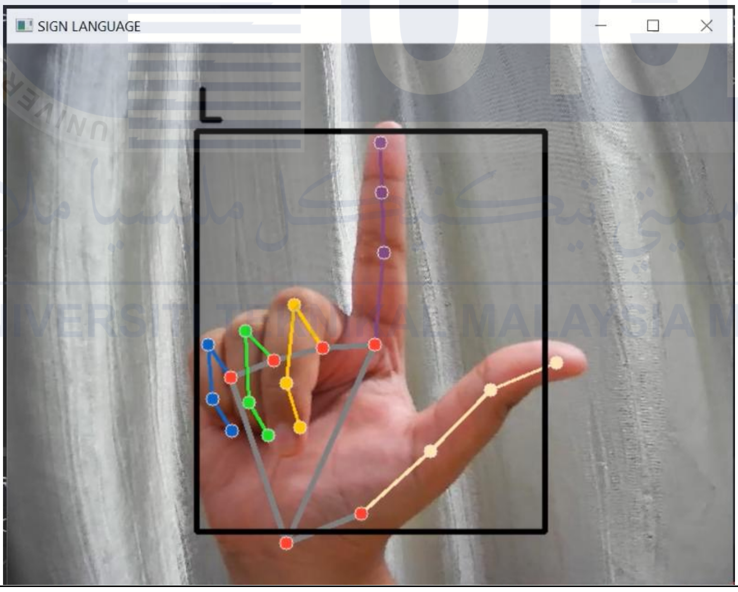
	A
	B

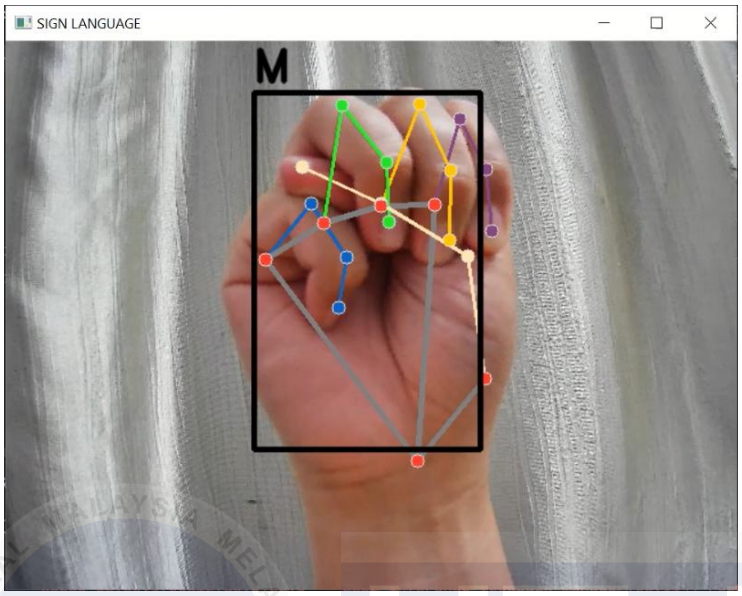
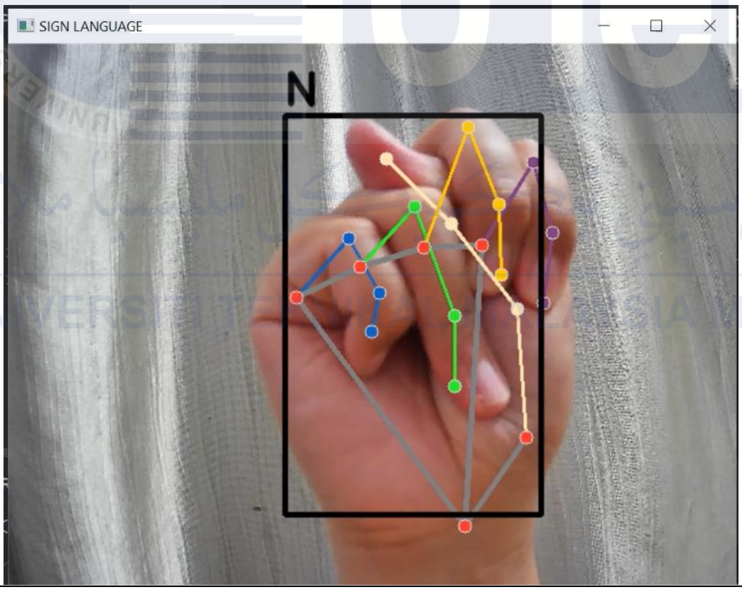
	C
	D

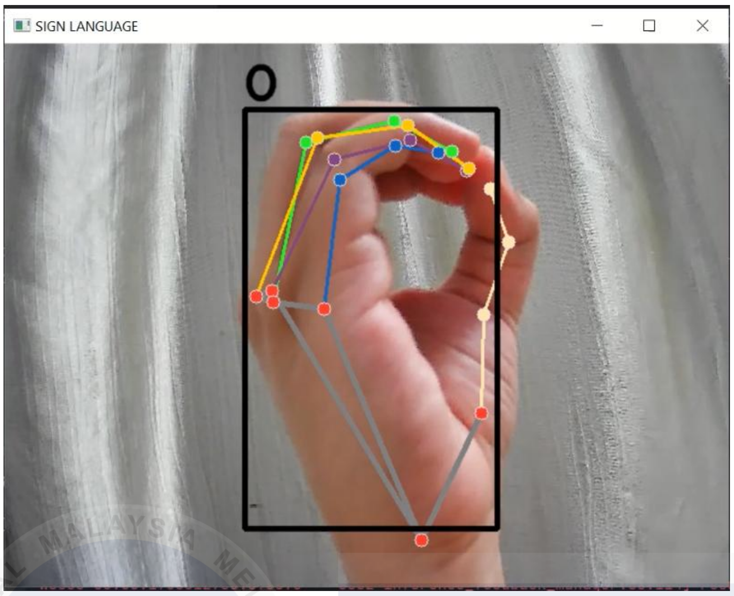
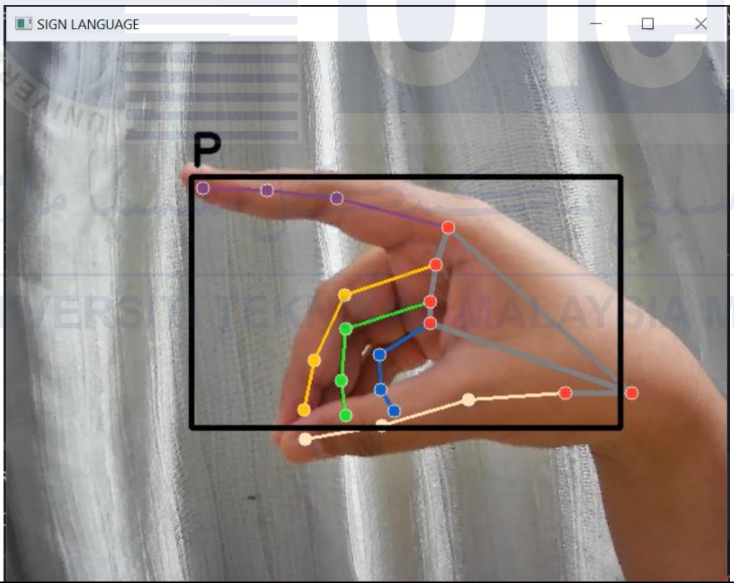
 SIGN LANGUAGE	E
 SIGN LANGUAGE	F

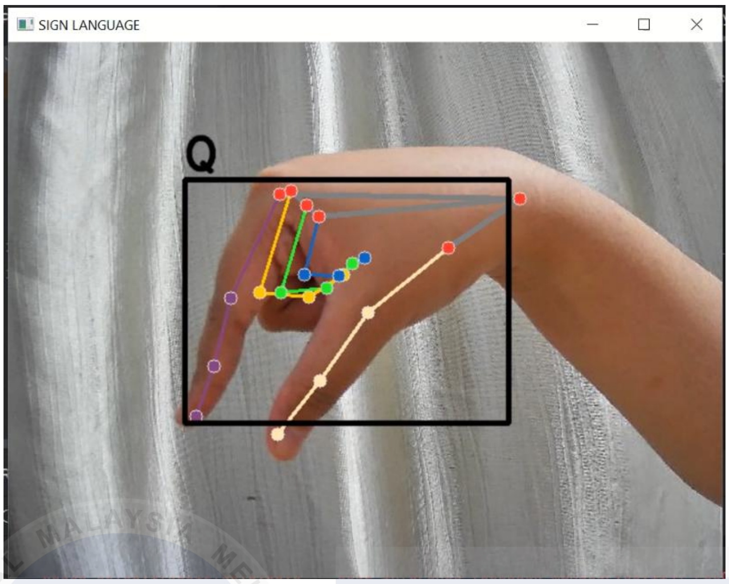
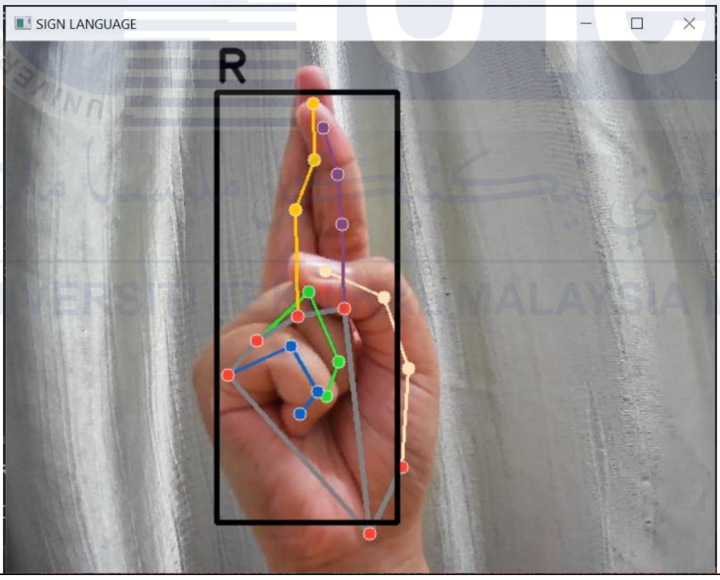
	G
	H

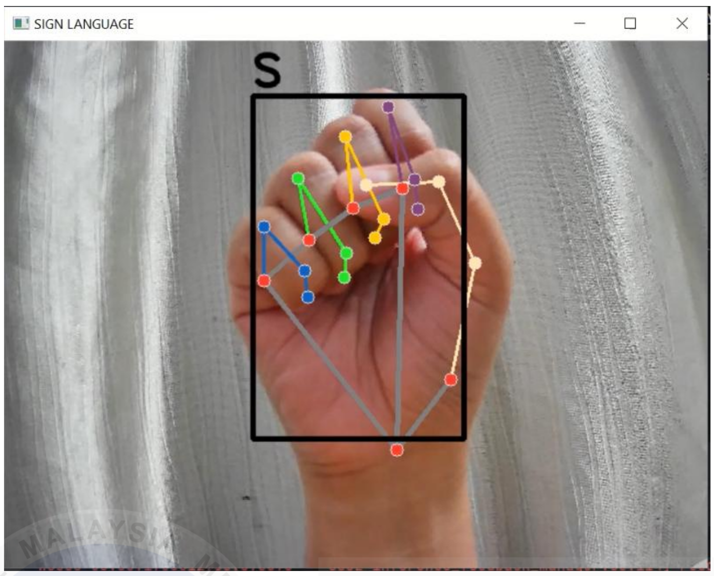
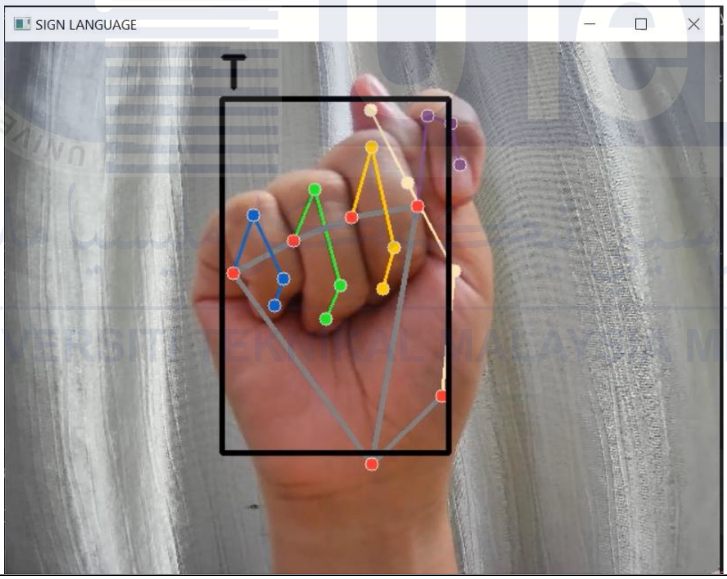
	I
	J

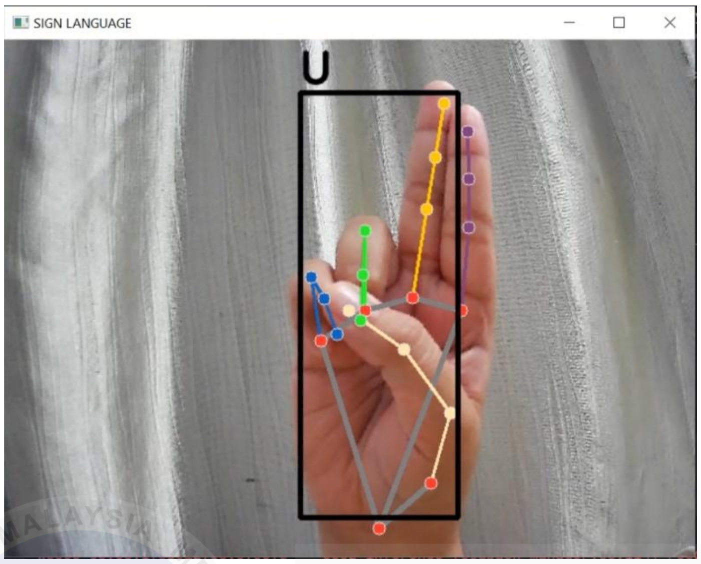
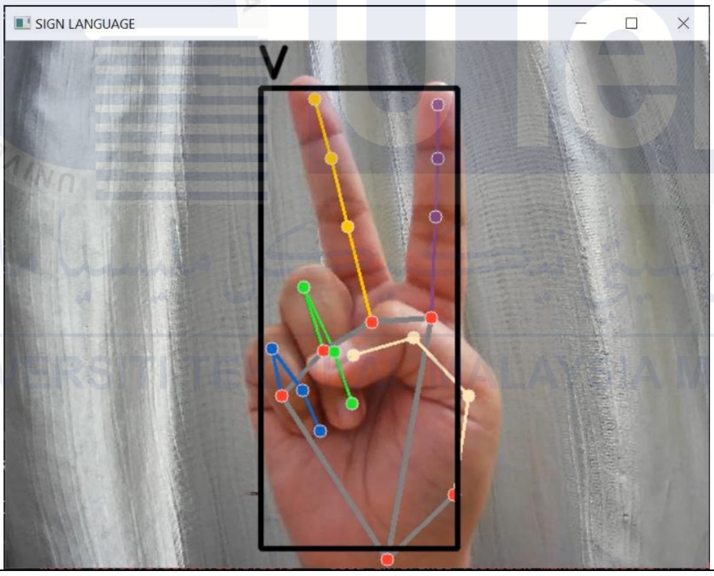
	K
	L

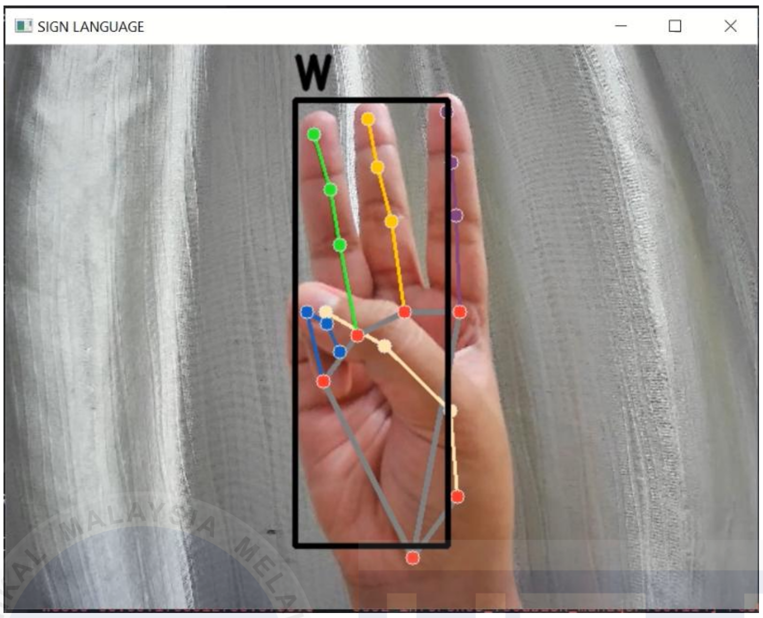
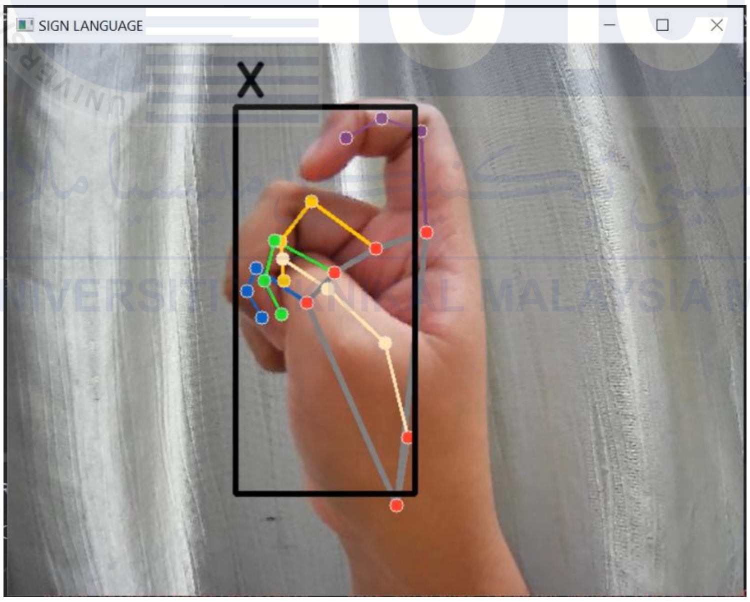
	M
	N

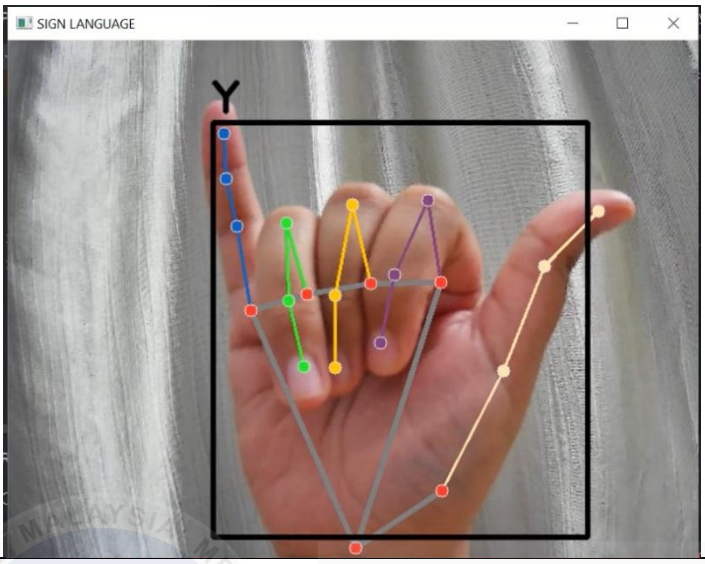
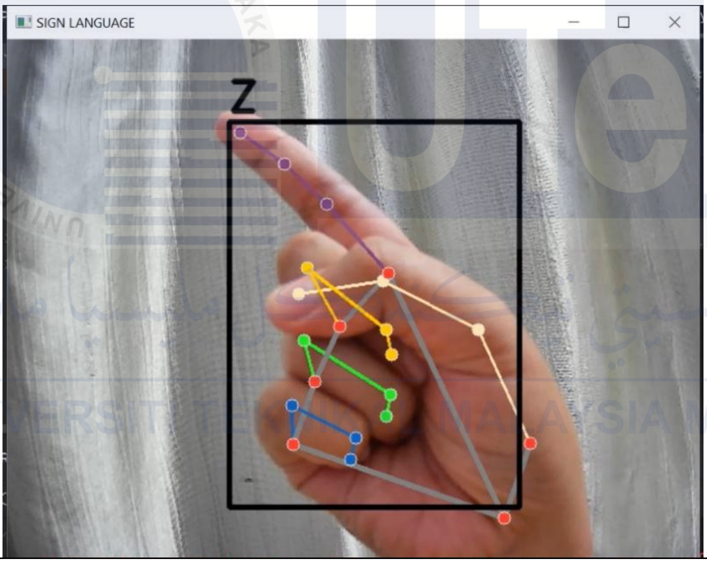
	O
	P

 SIGN LANGUAGE	Q
 SIGN LANGUAGE	R

	S
	T

	U
	V

	W
	X

 SIGN LANGUAGE	Y
 SIGN LANGUAGE	Z

4.6 Challenges and Limitations

Overall, the success of the sign language recognition system was good, but several challenges were encountered. A big problem was that not all gestures were unique, for instance, gestures that shared the same features, overlapping slightly, leading to occasional misclassification. For instance, the 'M' and 'N' letter gestures had identical hand configurations, which the model had a hard time differentiating between them. The sensitivity of the system to environmental factors (e.g., lighting and hand positioning) was another limitation because it robbed the system of consistency of landmark detection. The modeled touches did not always perform accurately because of changes in lighting conditions or small changes in hand position during the gesture execution. Such challenges point to areas of improvement that will help to increase the system robustness and accuracy.

4.7 Summary

This chapter presented sign language detection and identification the goal of this is to solve this gap and provide a platform from which it is easy to communicate in sign language and also learn it. The system it produced used Python and OpenCV to design a computer vision-based method to detect and recognize gestures in sign language. The successful implementation and high accuracy of the model highlight the potential for practical applications to improve communication and accessibility in the deaf and hard-of-hearing community.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the development of sign language detection and identification based on Python offers significant benefits in allowing individuals to learn and gain knowledge about sign language. By using this technology can be integrated into real-time translation applications, making it easier for people to understand each other, regardless of their language abilities. The research and development goals are to implement a system based on computer vision built by Python and OpenCV for the recognition and detection of sign language movements. The project aims to enhance the knowledge and understanding of sign language and improve the communication barrier between normal people and deaf people. The successful development and implementation of this system have the potential to positively impact the lives of people with deafness, enabling them to be confident in communicating with other people.

5.2 Potential for Commercialization

The potential for commercialization of sign language detection and identification technology is substantial. With the assistance of this technology, assistive devices for the deaf and hard-of-hearing community could be developed, such as, smartphone apps, and software applications with real-time sign language translation capabilities, which would enhance communication between sign language users and non-signers. This technology offers opportunities to develop interactive materials that support sign language education and instruction. Student learning experiences might be improved by using such educational software or applications. Additionally, the system has potential applications in the healthcare and rehabilitation fields. It can help with better communication between medical staff and patients who are hard of hearing or deaf. In public services and government, these systems can be particularly effective. Equipping public service kiosks and information booths with sign-language recognition systems can significantly help users access essential services. With accessibility standards and regulations, prioritizing user-friendliness and ease of integration, and establishing scalability and cost-effectiveness for mass deployment.

5.3 Future Works

The future works in the development of sign language detection and identification for deafness people can focus on several key areas such as:

i. Multi-hand identification and complex gestures such as enhance the model's ability to identify two-handed movements.

ii. The combination of other technologies to increase accessibility for hearing people such as voice synthesis apps like Convert Detected Signs into Spoken Language.

iii. Build a mobile app to make the system more efficenct for educational and assitive purpose.

REFERENCES

[1] World Health Organisation, “Deafness and hearing loss,” WHO, Feb. 02, 2024. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] National Geographic Society, “Sign Language,” education.nationalgeographic.org, May 20, 2022. https://education.nationalgeographic.org/resource/sign-language/

[3] I. DAYAS, “The history of sign language,” History, May 28, 2019. https://www.nationalgeographic.com/history/history-magazine/article/creation-of-sign-language

[4] “Gesture Recognition,” www.virtusa.com. https://www.virtusa.com/digital-themes/gesture-recognition

[5] M. Oudah, A. Al-Naji, and J. Chahl, “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques,” Journal of Imaging, vol. 6, no. 8, p. 73, Jul. 2020, doi: https://doi.org/10.3390/jimaging6080073.

[6] “Hand landmarks detection guide | Edge,” *Google for Developers*. https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

[7] “What is Sklearn? | Domino Data Lab,” *domino.ai*. https://domino.ai/data-science-dictionary/sklearn

[8] Wikipedia Contributors, “Random forest,” *Wikipedia*, Apr. 09, 2019. https://en.wikipedia.org/wiki/Random_forest

[9] A. Deep, A. Litoriya, A. Ingole, V. Asare, S. M. Bhole, and S. Pathak, “Realtime Sign Language Detection and Recognition,” in 2022 2nd Asian Conference on Innovation in Technology, ASIANCON 2022, 2022. doi: 10.1109/ASIANCON55314.2022.9908995.

[10] H. Adhikari, M. S. Bin Jahangir, I. Jahan, M. S. Mia, and M. R. Hassan, "A Sign Language Recognition System for Helping Disabled People," 2023 5th Int. Conf. Sustain. Technol. Ind. 5.0, STI 2023, vol. 0, pp. 1–6, 2023, doi: 10.1109/STI59863.2023.10465011.

[11] R. Yeware, A. Gokhale, S. Chalse, M. Samdekar, and J. Mante, "American Sign Language Detection using Deep Learning," 2023 7th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2023, pp. 1–6, 2023, doi: 10.1109/ICCUBEA58933.2023.10392208

[12] C. U. Om Kumar, K. P. K. Devan, P. Renukadevi, V. Balaji, A. Srinivas, and R. Krithiga, "Real Time Detection and Conversion of Gestures to Text and Speech to Sign System," 3rd Int. Conf. Electron. Sustain. Commun. Syst. ICESC 2022 - Proc., no. Icesc, pp. 73–78, 2022, doi: 10.1109/ICESC54411.2022.9885562

[13] P. Sykora, P. Kamencay, and R. Hudec, "Comparison of SIFT and SURF Methods for Use on Hand Gesture Recognition based on Depth Map," *AASRI Procedia*, vol. 9, pp. 19–24, 2014, doi: https://doi.org/10.1016/j.aasri.2014.09.005.

[14] T. A. Siby, S. Pal, J. Arlina, and S. Nagaraju, "Gesture based Real-Time Sign Language Recognition System," Proc. 2022 Int. Conf. Connect. Syst. Intell. CSI 2022, pp. 1–6, 2022, doi: 10.1109/CSI54720.2022.9924024

[15] C. Mummadi *et al.*, "Real-Time and Embedded Detection of Hand Gestures with an IMU-Based Glove," *Informatics*, vol. 5, no. 2, p. 28, Jun. 2018, doi: https://doi.org/10.3390/informatics5020028.

[16] P. Verma and K. Badli, "Real-Time Sign Language Detection using TensorFlow, OpenCV and Python," Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, no. 5, 2022, doi: 10.22214/ijraset.2022.43439.

[17] A. Mukhopadhyay, A. Chakrabarty, A. A. Das, and A. Sarkar, “Hand Gesture Based Recognition System,” 2023 7th Int. Conf. Electron. Mater. Eng. Nano-Technology, IEMENTech 2023, pp. 1–5, 2023, doi: 10.1109/IEMENTech60402.2023.10423522.

[18] S. Ameen and S. Vadera, “A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images,” *Expert Systems*, vol. 34, no. 3, p. e12197, Feb. 2017, doi: https://doi.org/10.1111/exsy.12197.

[19] S. Srivastava, A. Gangwar, R. Mishra, and S. Singh, “Sign Language Recognition System Using TensorFlow Object Detection API,” Commun. Comput. Inf. Sci., vol. 1534 CCIS, pp. 634–646, 2022, doi: 10.1007/978-3-030-96040-7_48.

[20] S. Kandasamy, S. Mowlieshwaran, R. Kiran, D. S. Kishore, and M. Karthikeyan, “Sign Language Recognition using Python and OpenCV,” Proc. - 2023 3rd Int. Conf. Smart Data Intell. ICSMDI 2023, pp. 88–92, 2023, doi: 10.1109/ICSMDI57622.2023.00023.

[21] A. Ganpatye and S. Mane, “Motion Based Indian Sign Language Recognition using Deep Learning,” 2022 2nd Int. Conf. Intell. Technol. CONIT 2022, pp. 1–6, 2022, doi: 10.1109/CONIT55038.2022.9848275.

[22] N. Kumar and A. K. Singh Bisht, “Hand sign detection using deep learning single shot detection technique,” ViTECoN 2023 - 2nd IEEE Int. Conf. Vis. Towar. Emerg. Trends Commun. Netw. Technol. Proc., pp. 1–7, 2023, doi: 10.1109/ViTECoN58111.2023.10157550.

[23] Thiluckraj G K and Dr. M. N. Nachappa, “Sign Language Recognition by Image Processing,” Int. J. Adv. Res. Sci. Commun. Technol., vol. 11, no. 6, pp. 306–310, 2024, doi: 10.48175/ijarsct-15954.

[24] “American Sign Language Dataset,” *www.kaggle.com*. https://www.kaggle.com/datasets/ayuraj/asl-dataset

[25] G. Lawton, "Data Preprocessing: Definition, Key Steps and Concepts," *SearchDataManagement*, Jan. 2022. https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing

[26] "6.2. Feature extraction," *scikit-learn*, 2018. https://scikit-learn.org/1.5/modules/feature_extraction.html# (accessed Jan. 06, 2025).

[27] Wikipedia Contributors, "Object detection," *Wikipedia*, Jul. 27, 2019. https://en.wikipedia.org/wiki/Object_detection

[28] GfG, "What is Image Recognition?," *GeeksforGeeks*, Feb. 13, 2024. https://www.geeksforgeeks.org/what-is-image-recognition/

[29] "The Importance of Accuracy in Machine Learning: A Comprehensive Guide," *www.artsyltech.com*. https://www.artsyltech.com/blog/Accuracy-In-Machine-Learning

[30] T. Srivastava, "Evaluation Metrics Machine Learning," *Analytics Vidhya*, Aug. 06, 2019. https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

[31] R. Kundu, "Confusion Matrix: How To Use It & Interpret Results [Examples]," *www.v7labs.com*, Sep. 13, 2022. https://www.v7labs.com/blog/confusion-matrix-guide

[32] Wikipedia Contributors, "Precision and recall," *Wikipedia*, Apr. 19, 2019. https://en.wikipedia.org/wiki/Precision_and_recall

[33] GeeksforGeeks, "F1 Score in Machine Learning," *GeeksforGeeks*, Dec. 27, 2023. https://www.geeksforgeeks.org/f1-score-in-machine-learning/

[34] Coursera, “What is Python used for? A beginner’s guide,” Coursera, Sep. 22, 2021. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[35] OpenCV, “About OpenCV,” OpenCV, 2018. https://opencv.org/about/

[36] “What is MediaPipe? A Guide for Beginners,” *Roboflow Blog*, Apr. 10, 2024. https://blog.roboflow.com/what-is-mediapipe/

APPENDICES

Appendix A

Appendix A Gantt Chart PSM 1

ACTIVITY	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Confirmation project's title	PSM Briefing and The Registration	🟪					Mid TermBreak							
Introduction (Chapter 1)		🟪	🟪	🟪										
Project progress						🟪				🟪	🟪	🟪		
Update Logbook					🟪	🟪					🟪	🟪		
Research journals (Literature review)			🟪	🟪	🟪	🟪		🟪	🟪	🟪	🟪	🟪	🟪	
Methodology (Chapter 3)									🟪	🟪				
Preliminary result analysis												🟪	🟪	
Full report progress												🟪	🟪	
Presentation PSM 1														🟪

Appendix B

Appendix B Gantt Chart PSM 2

ACTIVITY	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Development Feature extraction	▨	▨						Mid TermBreak						
Model testing		▨	▨	▨			▨							
Performance evaluation						▨	▨			▨	▨	▨		
Update Logbook						▨				▨				
Result and Discussion (Chapter 4)					▨	▨	▨		▨	▨	▨	▨		
Testing and Validation							▨		▨	▨				
Result analysis							▨			▨	▨	▨	▨	
Presentation PSM 2													▨	
Submission of final report														▨

APPENDICES

Appendix C Data Preprocessing Code

```
import os
import mediapipe as mp
import cv2
import pickle
import matplotlib.pyplot as plt

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

DATA_DIR = r'C:\DataAllSLALN'

data = []
labels = []

images = []

for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []
        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x)
                    data_aux.append(y)

                data.append(data_aux)
                labels.append(dir_)

f = open('data.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

APPENDICES

Appendix D Train Model Code

```
import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import numpy as np
import matplotlib.pyplot as plt  # Import Matplotlib for plotting

# Specify the path to your pickle file
pickle_file_path = 'C:\\PyCharm\\Project\\ProjectSL\\data.pickle'

# Load the data
with open(pickle_file_path, 'rb') as file:
   data_dict = pickle.load(file)

# Print the structure of the data to understand its shape
print("Data structure:", type(data_dict['data']), len(data_dict['data']))
print("Sample data:", data_dict['data'][:5])  # Print first 5 samples for inspection

# Assuming data_dict['data'] contains lists of varying lengths
data = data_dict['data']
labels = data_dict['labels']

# Filter out samples with a length of 84
filtered_data = []
filtered_labels = []

for sample, label in zip(data, labels):
    if len(sample) != 84:
        filtered_data.append(sample)
        filtered_labels.append(label)

# Convert filtered lists back to numpy arrays
filtered_data = np.array(filtered_data, dtype=object)  # Use dtype=object for
inhomogeneous data
filtered_labels = np.asarray(filtered_labels)

# Check class distribution
unique, counts = np.unique(filtered_labels, return_counts=True)
class_distribution = dict(zip(unique, counts))
print("Class distribution:", class_distribution)
```

```
# Split the filtered data into training and testing sets (80:20)
x_train, x_test, y_train, y_test = train_test_split(filtered_data, filtered_labels,
test_size=0.2, shuffle=True, stratify=filtered_labels)

# Initialize the Random Forest model with class weights
model = RandomForestClassifier(class_weight='balanced')

# Perform cross-validation on the training set
cv_scores = cross_val_score(model, list(x_train), y_train, cv=5)  # 5-fold cross-
validation

# Print cross-validation results
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score: {:.2f}%".format(np.mean(cv_scores) * 100))

# Plotting the cross-validation scores
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(cv_scores) + 1), cv_scores, marker='o', linestyle='-',
color='b')
plt.title('Cross-Validation Scores')
plt.xlabel('Fold Number')
plt.ylabel('Accuracy')
plt.ylim(0, 1)  # Set y-axis limits to [0, 1]
plt.xticks(range(1, len(cv_scores) + 1))  # Set x-ticks to be the fold numbers
plt.grid(True)
plt.show()

# Train the model on the entire training set
model.fit(list(x_train), y_train)

# Make predictions on the test set
y_test_predict = model.predict(list(x_test))

# Calculate accuracy on the test set
test_accuracy = accuracy_score(y_test, y_test_predict)
print('Test set accuracy: {:.2f}%'.format(test_accuracy * 100))

# Calculate Number of Correct Predictions and Total Number of Predictions
correct_predictions = (y_test_predict == y_test).sum()  # Count correct
predictions
total_predictions = len(y_test)  # Total number of predictions
```

APPENDICES

```
# Print the results
print('Number of Correct Predictions:', correct_predictions)
print('Total Number of Predictions:', total_predictions)

# Print additional metrics with zero_division parameter
print("Classification Report:\n", classification_report(y_test, y_test_predict,
zero_division=0))

# Set NumPy print options to display the full array
np.set_printoptions(threshold=np.inf, linewidth=np.inf)

# Print the confusion matrix in full view
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_predict))

# Save the model
with open('model.p', 'wb') as f:
    pickle.dump({'model': model}, f)
```

APPENDICES

Appendix E Test Model Code

```
import pickle
import cv2
import mediapipe as mp
import numpy as np

model_dict = pickle.load(open(r'C:\PyCharm\Project\ProjectSL\model.p', 'rb'))
model = model_dict['model']

# Open the video capture (0 for the default camera, or change to the appropriate
index)
cap = cv2.VideoCapture(1)
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True,
min_detection_confidence=0.3)

labels_dict = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9', 10:
'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F', 16: 'G', 17: 'H', 18: 'I', 19: 'J', 20: 'K',
21: 'L', 22: 'M', 23: 'N', 24: 'O', 25: 'P', 26: 'Q', 27: 'R', 28: 'S', 29: 'T', 30: 'U', 31:
'V', 32: 'W', 33: 'X', 34: 'Y', 35: 'Z'}
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame")
        break

    H, W, _ = frame.shape
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)

    data_aux = []
    x_ = []
    y_ = []
```

```
   if results.multi_hand_landmarks:
       for hand_landmarks in results.multi_hand_landmarks:
           mp_drawing.draw_landmarks(
               frame,
               hand_landmarks,
               mp_hands.HAND_CONNECTIONS,
               mp_drawing_styles.get_default_hand_landmarks_style(),
               mp_drawing_styles.get_default_hand_connections_style())

           for landmark in hand_landmarks.landmark:
               x = landmark.x
               y = landmark.y
               x_.append(x)
               y_.append(y)
               data_aux.append(x)
               data_aux.append(y)

       # Ensure we have the correct number of features
       if len(data_aux) == 42:  # 21 landmarks * 2 (x, y)
           # Make the prediction
           prediction = model.predict([np.asarray(data_aux)])
           prediction_character = labels_dict[int(prediction[0])]

           # Calculate bounding box coordinates
           x1 = int(min(x_) * W) - 10
           y1 = int(min(y_) * H) - 10
           x2 = int(max(x_) * W) - 10
           y2 = int(max(y_) * H) - 10

           # Draw bounding box and prediction text
           cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
           cv2.putText(frame, prediction_character, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3,
                       cv2.LINE_AA)

    cv2.imshow('SIGN LANGUAGE', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break  # Correctly exit the loop when 'q' is pressed

# Release the video capture object and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

APPENDICES