# SIGN LANGUAGE DETECTION AND TRANSLATION USING PYTHON

# A PROJECT REPORT

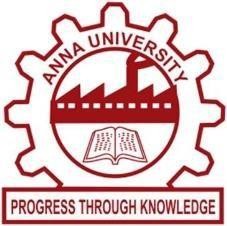
***Submitted by***

|  |  |
| --- | --- |
| V.SAAI MIRUTHANJAYAN | 2020504570 |
| R. SARAVANAPERUMAL | 2020504579 |
| T.SRI RAM | 2020504587 |

***In partial fulfilment for the award of the degree Of***

# BACHELOR OF ENGINEERING IN

**ELECTRONICS ENGINEERING**



# MADRAS INSTITUTE OF TECHNOLOGY ANNA UNIVERSITY, CHENNAI 600 025

**AUGUST 2022**

# TABLE OF CONTENTS;

**1.ACKNOWLEDGMENT**

**2. INTRODUCTION**

**3. OBJECTIVE**

**4. METHODOLOGY**

**5. ALPHABETS OF AMERICAN SIGN LANGUAGE**

**6.SOFTWARE USED**

**7.LIBRARIES USED**

**8.SOURCE CODE**

**9.CONCLUSION**

**10.REFERENCES**

# ACKNOWLEDGEMENT

We consider it as our privilege and our primary duty to express our gratitude and respect to all those who guided and inspired us in the successful completion of the project.

We owe solemn gratitude to **Dr. J. PRAKASH**, Dean, Madras Institute of Technology, for having given consent to carry out the project work at MIT Campus, Anna University.

We wish to express our sincere appreciation and gratitude to **Dr. M. GANESH MADHAN**, Professor and Head of the Department of Electronics Engineering, who has encouraged and motivated us in our endeavours.

We are extremely grateful to our project guide **Dr.M.MANIKANDAN** for their timely and thoughtful guidance and encouragement for the completion of the project.

We also thank all the teaching and non-teaching staff members of the Department of Electronics Engineering for their support in all aspects.

**INTRODUCTION**

This project helps us to detect or identify sign language and efficiently translate it in realtime using pycharm which is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web and data science development.

The purpose of sign language recognition systems is to provide an efficient and accurate way to convert sign language into text or voice. Recent field of research is intended to focus on effectively recognizing signs under computing power constraints.

Although sign language is the most natural way of exchanging information among deaf people it has been observed that they are facing difficulties with normal people interaction. Sign language consists of vocabulary of signs in exactly the same way as spoken language consists of a vocabulary of words. Sign languages are not standard and universal and the grammars differ from country to country .In this project we are using American Sign Language for detection and translation.

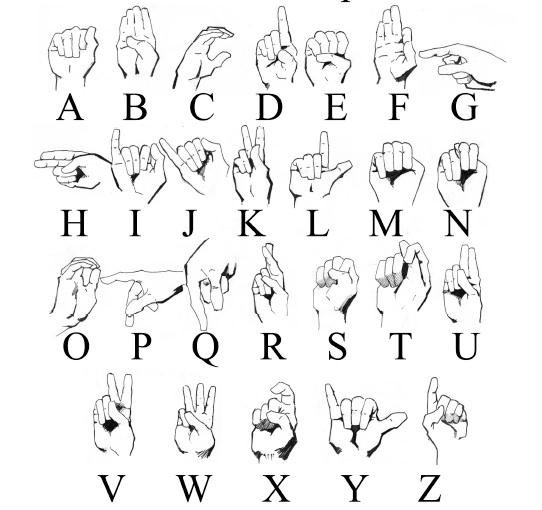
**OBJECTIVE:**

The main objective of the project is to demonstrate Detection and Translation of Sign language in real time using pycharm which is a Python Integrated Development Environment (IDE).

**METHODOLOGY:**

At first a basic study was done on sign language and we chose alphabets of American sign language as the samples for the detection and translation. For the samples for the detection and translation a python source code named **DATACOLLECTION** is used to capture multiple samples of the alphabets of sign language using a webcam. The collected samples are then used create a machine learning model. Teachable machine is web based tool used to convert the collected samples to a machine learning model. This model is then specified in another python source code named **DETECTION AND TRANSLATION** which is used in detection and translation of the alphabets of American sign language.

**ALPHABETS OF AMERICAN SIGN LANGUAGE**

****

**SOFTWARE USED:**

**PYCHARM:**

Pycharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web and data science development.

Pycharm is available in three editions

* **Community:** For Python development including code assistance, refactorings, visual debugging and version control integration. It is free and open sourced.
* **Professional:** For professional Python, web and data science development, including code assistance, refactorings, visual debugging and version control integration, remote configurations, deployment, support for popular web frameworks. It is paid to use.
* **Edu:** For learning programming languages and related technologies with integrated educational tools.

**LIBRARIES USED:**

* CVZONE
* MEDIAPIPE

**SOURCE CODE**

**1.CODE FOR DATACOLLECTION:**

import cv2  
from cvzone.HandTrackingModule import HandDetector  
import numpy as np  
import math  
import time  
  
cap = cv2.VideoCapture(0)  
detector = HandDetector(maxHands=1)  
  
offset = 20  
imgSize = 300  
  
folder = "Data/C"  
counter = 0  
  
while True:  
 success, img = cap.read()  
 hands, img = detector.findHands(img)  
 if hands:  
 hand = hands[0]  
 x, y, w, h = hand['bbox']  
  
 imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 255  
 imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]  
  
 imgCropShape = imgCrop.shape  
  
 aspectRatio = h / w  
  
 if aspectRatio > 1:  
 k = imgSize / h  
 wCal = math.ceil(k \* w)  
 imgResize = cv2.resize(imgCrop, (wCal, imgSize))  
 imgResizeShape = imgResize.shape  
 wGap = math.ceil((imgSize - wCal) / 2)  
 imgWhite[:, wGap:wCal + wGap] = imgResize  
  
 else:  
 k = imgSize / w  
 hCal = math.ceil(k \* h)  
 imgResize = cv2.resize(imgCrop, (imgSize, hCal))  
 imgResizeShape = imgResize.shape  
 hGap = math.ceil((imgSize - hCal) / 2)  
 imgWhite[hGap:hCal + hGap, :] = imgResize  
  
 cv2.imshow("ImageCrop", imgCrop)  
 cv2.imshow("ImageWhite", imgWhite)  
  
 cv2.imshow("Image", img)  
 key = cv2.waitKey(1)  
 if key == ord("o"):  
 counter += 1  
 cv2.imwrite(f'{folder}/Image\_{time.time()}.jpg', imgWhite)  
 print(counter)

**2.CODE FOR DETECTION AND TRANSLATION:**

import cv2  
from cvzone.HandTrackingModule import HandDetector  
from cvzone.ClassificationModule import Classifier  
import numpy as np  
import math  
  
cap = cv2.VideoCapture(0)  
detector = HandDetector(maxHands=1)  
classifier = Classifier("Model/keras\_model.h5", "Model/labels.txt")  
  
offset = 20  
imgSize = 300  
  
folder = "Data/C"  
counter = 0  
  
labels = ["A", "B", "C"]  
  
while True:  
 success, img = cap.read()  
 imgOutput = img.copy()  
 hands, img = detector.findHands(img)  
 if hands:  
 hand = hands[0]  
 x, y, w, h = hand['bbox']  
  
 imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 255  
 imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]  
  
 imgCropShape = imgCrop.shape  
  
 aspectRatio = h / w  
  
 if aspectRatio > 1:  
 k = imgSize / h  
 wCal = math.ceil(k \* w)  
 imgResize = cv2.resize(imgCrop, (wCal, imgSize))  
 imgResizeShape = imgResize.shape  
 wGap = math.ceil((imgSize - wCal) / 2)  
 imgWhite[:, wGap:wCal + wGap] = imgResize  
 prediction, index = classifier.getPrediction(imgWhite, draw=False)  
 print(prediction, index)  
  
 else:  
 k = imgSize / w  
 hCal = math.ceil(k \* h)  
 imgResize = cv2.resize(imgCrop, (imgSize, hCal))  
 imgResizeShape = imgResize.shape  
 hGap = math.ceil((imgSize - hCal) / 2)  
 imgWhite[hGap:hCal + hGap, :] = imgResize  
 prediction, index = classifier.getPrediction(imgWhite, draw=False)  
  
 cv2.rectangle(imgOutput, (x - offset, y - offset - 50),  
 (x - offset + 90, y - offset - 50 + 50), (255, 0, 255), cv2.FILLED)  
 cv2.putText(imgOutput, labels[index], (x, y - 26), cv2.FONT\_HERSHEY\_COMPLEX, 1.7, (255, 255, 255), 2)  
 cv2.rectangle(imgOutput, (x - offset, y - offset),  
 (x + w + offset, y + h + offset), (255, 0, 255), 4)  
  
 cv2.imshow("ImageCrop", imgCrop)  
 cv2.imshow("ImageWhite", imgWhite)  
  
 cv2.imshow("Image", imgOutput)  
 cv2.waitKey(1)

**CONCLUSION:**

The code for sign language detection and translation is successfully executed in pycharm using cvzone and mediapipe libraries.

**REFERENCES:**

* [**https://core.ac.uk/download/pdf/55638597.pdf**](https://core.ac.uk/download/pdf/55638597.pdf)
* [**https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/#:~:text=In%20this%20sign%20language%20recognition,and%20Keras%20modules%20of%20python**](https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/#:~:text=In%20this%20sign%20language%20recognition,and%20Keras%20modules%20of%20python)**.**
* [**https://www.analyticsvidhya.com/blog/2021/06/sign-language-recognition-for-computer-vision-enthusiasts/**](https://www.analyticsvidhya.com/blog/2021/06/sign-language-recognition-for-computer-vision-enthusiasts/)