# Hand Landmarks Detection

by

**Sumedha Saxena**

**1155209203**

A report

submitted in partial fulfilment

of the requirements for the degree of

Master of Science in Biomedical Engineering

The Chinese University of Hong Kong

Dec-23

Supervisor: Prof. Zheng Li

Sumedha Saxena
1155209203
BMEG5920

# Academic Honesty Declaration Statement

**The Chinese University of Hong Kong**
**Academic Honesty Declaration Statement**

Submission Details

| | | | |
|---|---|---|---|
| **Student Name** | SAXENA Sumedha (s1155209203) | | |
| **Year and Term** | 2023-2024 Term 1 | | |
| **Course** | BMEG-5920--J01 M.Sc. Project I | | |
| **Assignment Marker** | Professor HO Ho Pui | | |
| **Submitted File Name** | Hand Landmarks Detection - Progress report.docx | | |
| **Submission Type** | Individual | | |
| **Assignment Number** | 1 | **Due Date (provided by student)** | 2023-12-30 |
| **Submission Reference Number** | 3868342 | **Submission Time** | 2023-12-30 16:14:03 |

Agreement and Declaration on Student's Work Submitted to VeriGuide

VeriGuide is intended to help the University to assure that works submitted by students as part of course requirement are original, and that students receive the proper recognition and grades for doing so. The student, in submitting his/her work ("this Work") to VeriGuide, warrants that he/she is the lawful owner of the copyright of this Work. The student hereby grants a worldwide irrevocable non-exclusive perpetual licence in respect of the copyright in this Work to the University. The University will use this Work for the following purposes.

(a) Checking that this Work is original

The University needs to establish with reasonable confidence that this Work is original, before this Work can be marked or graded. For this purpose, VeriGuide will produce comparison reports showing any apparent similarities between this Work and other works, in order to provide data for teachers to decide, in the context of the particular subjects, course and assignment. In addition, the Work may be investigated by AI content detection software to determine originality. However, any such reports that show the author's identity will only be made available to teachers, administrators and relevant committees in the University with a legitimate responsibility for marking, grading, examining, degree and other awards, quality assurance, and where necessary, for student discipline.

(b) Anonymous archive for reference in checking that future works submitted by other students of the University are original

The University will store this Work anonymously in an archive, to serve as one of the bases for comparison with future works submitted by other students of the University, in order to establish that the latter are original. For this purpose, every effort will be made to ensure this Work will be stored in a manner that would not reveal the author's identity, and that in exhibiting any comparison with other work, only relevant sentences/ parts of this Work with apparent similarities will be cited. In order to help the University to achieve anonymity, this Work submitted should not contain any reference to the student's name or identity except in designated places on the front page of this Work (which will allow this information to be removed before archival).

(c) Research and statistical reports

The University will also use the material for research on the methodology of textual comparisons and evaluations, on teaching and learning, and for the compilation of statistical reports. For this purpose, only the anonymously archived material will be used, so that student identity is not revealed.

I confirm that the above submission details are correct. I am submitting the assignment for:

[ X ] an individual project.

I have read the above and in submitting this Work fully agree to all the terms. I declare that: (i) the assignment here submitted is original except for source material explicitly acknowledged; (ii) the piece of work, or a part of the piece of work has not been submitted for more than one purpose (e.g. to satisfy the requirements in two different courses) without declaration; and (iii) the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I also acknowledge that I am aware of the University's policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website http://www.cuhk.edu.hk/policy/academichonesty/.

I declare that I have not distributed/ shared/ copied any teaching materials without the consent of the course teacher(s) to gain unfair academic advantage in the assignment/ course.

I declare that I have read and understood the University's policy on the use of AI for academic work. I confirm that I have complied with the instructions given by my teacher regarding the use of AI tools for this assignment and consent to the use of AI content detection software to review my submission.

I also understand that assignments without a properly signed declaration by the student concerned will not be graded by the teacher(s)

Signature (SAXENA Sumedha, s1155209203)

30/12/2023

Date

Instruction for Submitting Hard Copy / Soft Copy of the Assignment

This signed declaration statement should be attached to the hard copy assignment or submission to the course teacher, according to the instructions as stipulated by the course teacher. If you are required to submit your assignment in soft copy only, please print out a copy of this signed declaration statement and hand it in separately to your course teacher.

Sumedha Saxena
1155209203
BMEG5920

## 1. Abstract

The aim of the project 'Hand landmarks detection' is to detect and highlight the location of joints in the image of a hand. Along with the location, this projects also calculates the distance been the joints (landmarks) in pixels as well as metric units.

The solution harnesses the power of Machine Learning with Convolutional Neural Network for feature extraction from images and the use of OpenCV library to manipulate the image for precise distance calculation.

The motivation for the development of this project is to help with the rehabilitation program for stroke patients, who have suffered loss in motor functions, especially in the movement of hand or fingers.

As a part of this rehabilitation program, these patients are provided with a robotic glove that they wear on the affected hand. This glove helps facilitates movement and provides training to slowly regain activity in the affected areas.

Since different patients have different hand sizes, they need gloves with a custom fit for best results. Getting the measurement right manually can be a tedious and time-consuming process.

This project attempts at making the process of getting measurements for each patient, in a faster and efficient fashion, with least manual intervention. It leverages Machine Learning techniques and Image processing methodologies to calculate the distance between the joints for different patients, as accurately as possible.

To start with the process, the image of the back of the hand of patient is taken with an A4 sheet of paper in the background.
This serves as an input for this project. The project then uses open-source machine learning model to identify the key points along with the handedness(left/right) of the hand.
The output of the machine leaning model is the same image with the identified landmarks drawn on top of the image, along with the pixel coordinates for each identified joint location.

The project then further uses image processing techniques to calculate the pixel distance between the joints and then convert them to metric units for real world usage and applications. The calculated distance is displayed on the original image at appropriate locations.

For testing, a dataset of hands was collected from healthy individuals as well as stroke-affected patients, and the solution was able to accurately identify the key-points for all test images and was able to calculate the distance between the identified key-points.

Sumedha Saxena
1155209203
BMEG5920

# Table of Contents




Sumedha Saxena
1155209203
BMEG5920

## 2. Background and Motivation

The motivation for this project is to help the team at CUHK that works with development of soft robotic hands/gloves for stroke survivors.

Stroke is a cerebral blood circulation disorder that leads to sudden loss of brain function. As per the studies, around 85% of stroke patients worldwide have hand dysfunction, with 60% still suffering from upper limb disorders in fingers and wrists. (Bo Sheng, 2023)

As a part of rehabilitation program, CUHK team works with development of robotic glove that helps with movement of hand and fingers, to restore the original function gradually with training.

Since the size if the hands differ for each patient, we could use a tool that can automatically detect the joints in the hands and calculate the distance between important joints for precise development of the robotic glove.

This project aims at helping to make that process automated and efficient by leveraging machine learning and image processing technology, to calculate right size for different patients.

## 3. Research and Study

To understand how the aim of the project can be achieved and what are available options for fastest and most efficient results, following topics were researched.

### 3.1 Study deep learning and Neural Network

Since the aim of the project is to identify patterns from an image, Machine Learning is our area of interest. It is an area of Artificial Intelligence concerned with the development of algorithms that can perform tasks without explicit instructions.

For tasks that require image classification, object recognition, and image segmentation, Deep Neural Networks like CNN have been instrumental in achieving state of the art performance. Therefore, studying the behaviour of Convolutional Neural Networks was an important step for this project. (Liu, 2018)

### 3.2 Study Python libraries and toolkits

From the study of neural networks, it was realised that Python is a language of choice when it comes to implementation of machine learning solutions, because of the large availability of libraries and toolkits that are provided for faster development.

Below is the list of 2 libraries that were looked at for our development purpose:

- **PyTorch**

PyTorch is an open-source machine learning Python library that's based on the C programming language framework, Torch. PyTorch qualifies as a data science library and can integrate with other similar Python libraries such as NumPy. (Gupta, 2022)

PyTorch is known for its high speeds of execution even when it's handling heavy and extensive graphs. It's also highly flexible, which allows it to operate on simplified processors in addition to CPUs and GPUs. PyTorch comes with a collection of powerful APIs that lets you expand on the PyTorch library, as well as a natural language toolkit for smoother processing. It's compatible with Python's IDE tools, which makes for an easy debugging process.

- **TensorFlow**

TensorFlow is a free and open-source Python library that specializes in differentiable programming. The library offers a collection of tools and resources that help make building DL and ML models and neural networks straightforward for beginners and professionals.

TensorFlow can be used to implement reinforcement-learning in ML and DL models and allows you to directly visualize your machine learning models with its built-in tools. (Gupta, 2022)

## 3.3 Identify existing open-source solutions.

Implementing a CNN model from scratch and train it to correctly identify landmarks from the image of hand was possible using the Python libraries but it would require a lot of time to correctly train it. Along with the training, we would need a huge dataset of sample images to train the model with.

So further research was conducted to find existing solutions that might be catering to our need for identifying landmarks in the image of a hand.

Below are the details of 2 open-source solutions found that fit our purpose.

They been developed using machine learning, that can identify the joints in a hand using a static image or with real-time recognition via web-camera. But these tools do not calculate the distance between the identified landmarks.

- **MMPose:**

**Github location**: https://github.com/open-mmlab/mmpose/tree/main

**Summary**: MMPose is a Pytorch-based pose estimation open-source toolkit, a member of the OpenMMLab Project. It contains a wide variety of pre-trained models and set of algorithms for 2D hand pose and landmark detection along with detection. (hand-keypoint-estimation, n.d.)

For demo purposes we used quick Python implementation to run the program for the image of a hand that is a part of their existing dataset. With some basic configuration, the program returned the identified landmarks displayed on the original image.

Here is the result of the detection from MMPose library:

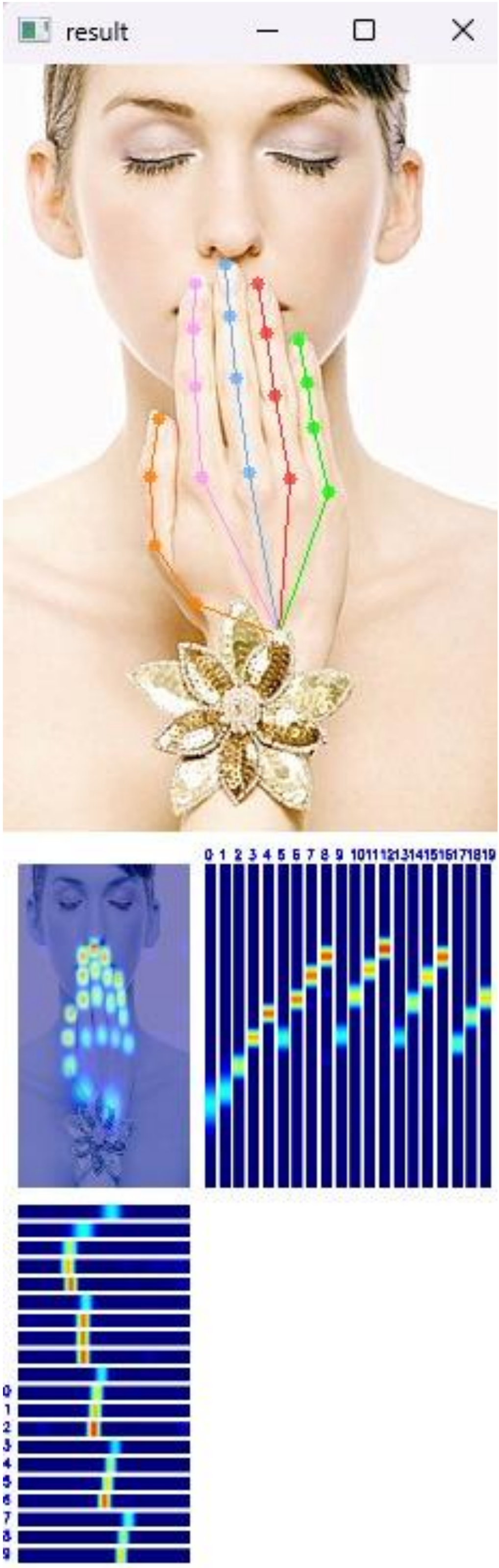



Fig: Result from MMPose model for hand landmark detection

This library also returns the coordinates of the identified key points in the hand, and no information regarding the pixel distance or the real-world distance between the landmarks that we needed for this project.

Sumedha Saxena
1155209203
BMEG5920

- **MediaPipe:**

**Github location**: https://github.com/google/mediapipe

**Summary:** MediaPipe is a cross-platform open-source library developed by Google that provides ready-to-use ML solutions for computer vision tasks. It contains a suite of libraries and tools to quickly apply artificial intelligence (AI) and machine learning (ML) techniques in applications.

The MediaPipe Hand Landmarker task lets you detect the landmarks of the hands in an image. You can use this task to locate key points of hands and render visual effects on them. This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates, hand landmarks in world coordinates and handedness (left/right hand) of multiple detected hands. (hand_landmarker, n.d.) (MediaPIpe Solutions guide, n.d.)

This tool also returns the pixel coordinates of the identified landmarks and no further information about the actual distance between the identified joints.

### 3.4 Study techniques for distance calculation between identified key points.

The open-source solutions mentioned above, do a huge deal of the task by identifying the key points, but we needed to write our own logic for the calculation of distance between the key points and displaying the result on the image.

This task would require carrying out multiple steps as a part of image processing like Image segmentations, thresholding, contouring etc. Hence, some working knowledge of image processing was required, and we needed a python-compatible library to easily carry out such operations.

**OpenCV:**

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms.

Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), colour space conversion, histograms, and so on. (Image Processing in OpenCV, n.d.)

## 4. Implementation details

After studying the documentation for both MMPose and MediaPipe solutions and running a demo, we decided to go ahead with **MediaPipe** for our implementation.

### 4.1 Tools/Technology used:

| **Language** | **Python** |
|---|---|
| **Image processing library** | OpenCV |

| | |
|---|---|
| **Open-source solution for CNN** | MediaPipe |

### 4.2 Logic:

Create a python console app that implements MediaPipe libraries and calls its methods to run the landmark detection on the image of a hand.

The MediaPipe library would display the identified landmarks on the input image and return the pixel coordinates of the identified landmarks.

Additionally, we write logic to get the pixel distance between the coordinates.

To convert the pixel distance into cms, we use an A4 sheet of paper in the background such that the hand is placed over the A4 sheet before clicking the picture.

Then, we write additional logic to get the pixel coordinates of the corners of A4 sheet and since we know the distance between the corners in cms, we can derive a scaling factor, for converting the pixel distance of joints in the same picture to cms.

Lastly, we display the calculated distance between joints on the image along with the identified landmarks.

### 4.3 Steps:

The implementation process can be roughly divided into following steps:

1. Import and use **MediaPipe** libraries for hand landmark detection and visualization.
    a. Create a Python console app and install the required **MediaPipe** libraries on a new virtual environment on the development machine.
    b. Specify the name of the model that MediaPipe would be using for the hand landmark detection.
    c. Specify the location on input image that we need to detect landmarks for.
    d. Call the MediaPipe utility to run the detection on the input image.
    e. Get and store the results from MediaPipe into the memory of the application. The result contains the following information:
        - ❖ Handedness of the hand in input image (Left/Right)
        - ❖ A collection of pixel coordinates identified for the 21 key points in the hand.
    f. The MediaPipe documentation contains instruction on how to draw the detected landmarks on the image itself, so using those instructions the detected key points are visualized on the original image.

2. **Get the pixel distance** between the joints.
    a. The result data from MediaPipe library contains pixel coordinates for the detected key points. Using that, we write the logic to get the pixel distance between 2 coordinates on the 2D space.
    b. This is done for the specific landmarks that we are interested in and would be useful for the robotic-glove development.
    c. Once we have the pixel distance, we use **OpenCV** library to write the pixel distance on the image at appropriate locations (mid-way between the 2 coordinates).

3. **Get the handedness** (left/right) of the hand.
    a. The result from MediaPipe library contains the handedness information. We use OpenCV library to write the returned handedness on the input image.

4. **Find the contours** of the A4 sheet paper in background and get the coordinates of the corners.
    a. To get a scaling factor, for conversion between pixel distance to metric units, we need to quantify the pixel distance between the corners of A4 sheet.
    b. We use OpenCV library to do following transformations to the image:
        - ❖ Convert the image to Grayscale.
        - ❖ Apply Gaussian blur to remove noise and smooth out the image.
        - ❖ Apply **thresholding** to the image. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. This is done to highlight only the white A4 sheet to maximum intensity and everything else to lowest intensity. Here is a sample result after thresholding of the input image.

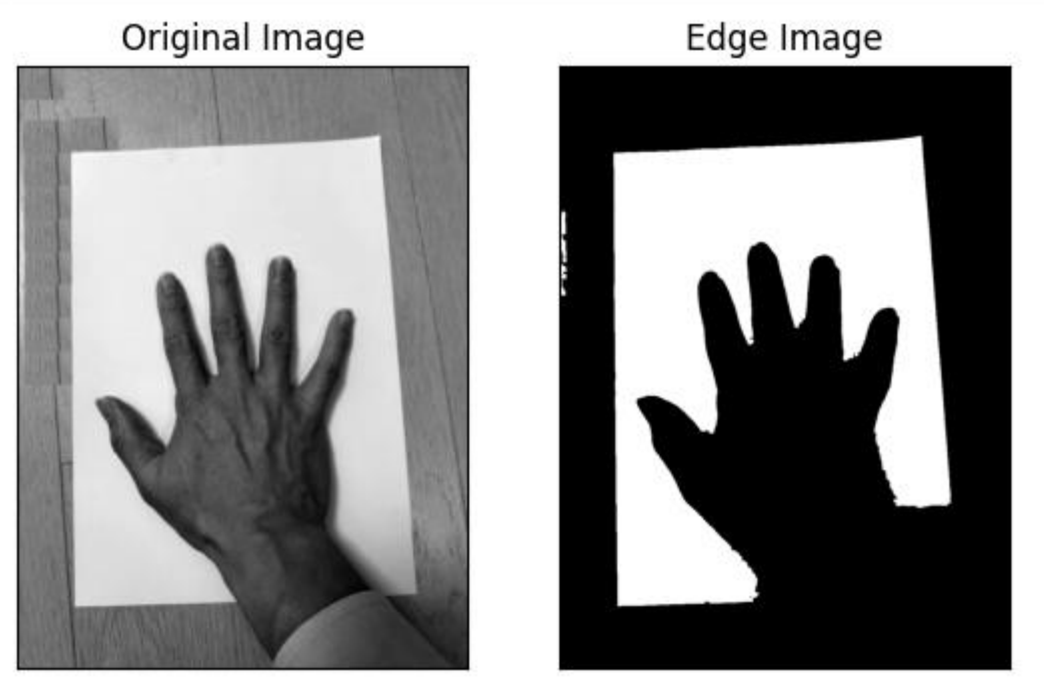



FIG: RESULT AFTER APPLICATION OF GAUSSIAN BLUR AND THRESHOLDING TO ORIGINAL GRAYSCALE IMAGE

- ❖ Next step is to **find the contours** of the A4 sheet of paper, from the resulting edge image using OpenCV library. The biggest contour is selected as the one we are interested in. The intermediate result of contouring, and selection of biggest contouring is shown below:

❖ 


FIG: RESULT AFTER FINDING CONTOURS IN THE IMAGE AND SELECTING THE BIGGEST CONTOUR

❖ The contours contain the pixel coordinates that we are interested in.

5. **Get the pixel distance of the A4 sheet** corners and get the 'scaling factor' with the distance in cms.
    a. We use logic to extract the contour coordinates belonging to the corners of A4 sheet of paper and get the pixel distance. Using our knowledge of the standard length of A4 sheet dimension in cms, we get a scaling factor to convert pixel distance into cms.

6. **Get the distance between the joints** of the hand by converting the pixel distance to cms, using the scaling factor.
    a. Using the obtained scaling factor, we go back to the saved pixel distance that was calculated from the MediaPipe library result dataset. The scaling factor is used to convert all pixel distances to cms.

7. **Display the length in cms** (rounded to 2 decimal places) on the image, along with the pixel distance for comparison.
    a. Once we have the distance between the hand landmarks in cms, we use the OpenCV library to display it on the original image along with detected landmarks. We do not display the identified contours. The contouring operation is done just to get the scaling factor using A4 sheet.

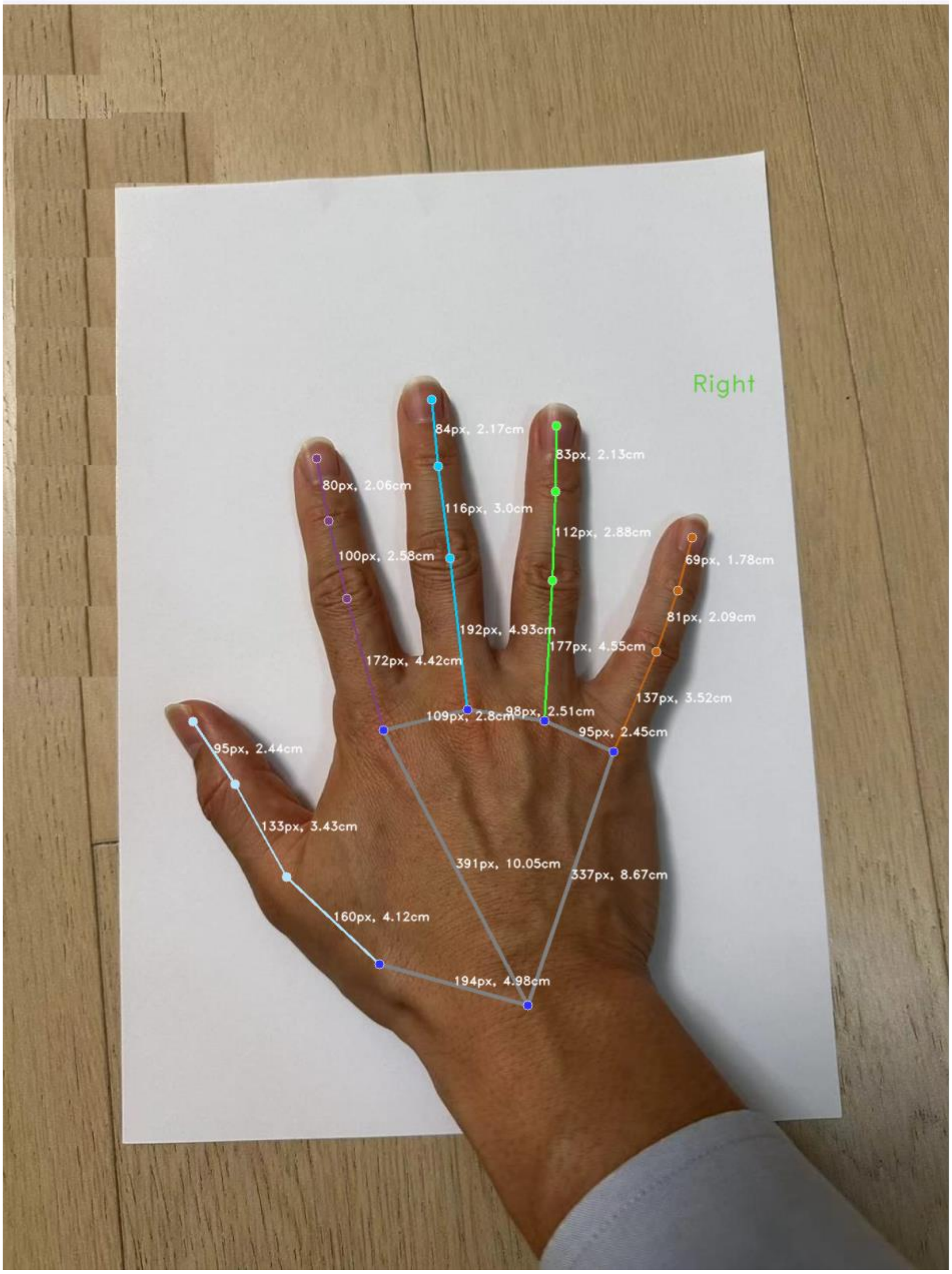
## 5. Preliminary results

From the current implementation we were able to achieve the following results:

a) Handedness (Left/Right) of the hand in the input image.
b) Identification and visualization of the joints in the hand
c) Distance between joints in pixel as well as cms, both displayed on the original image itself.

Testing was done with the images of hands of real people and the aforementioned information was retrieved for all of them.
Below are images showing the aforementioned results for different test images.

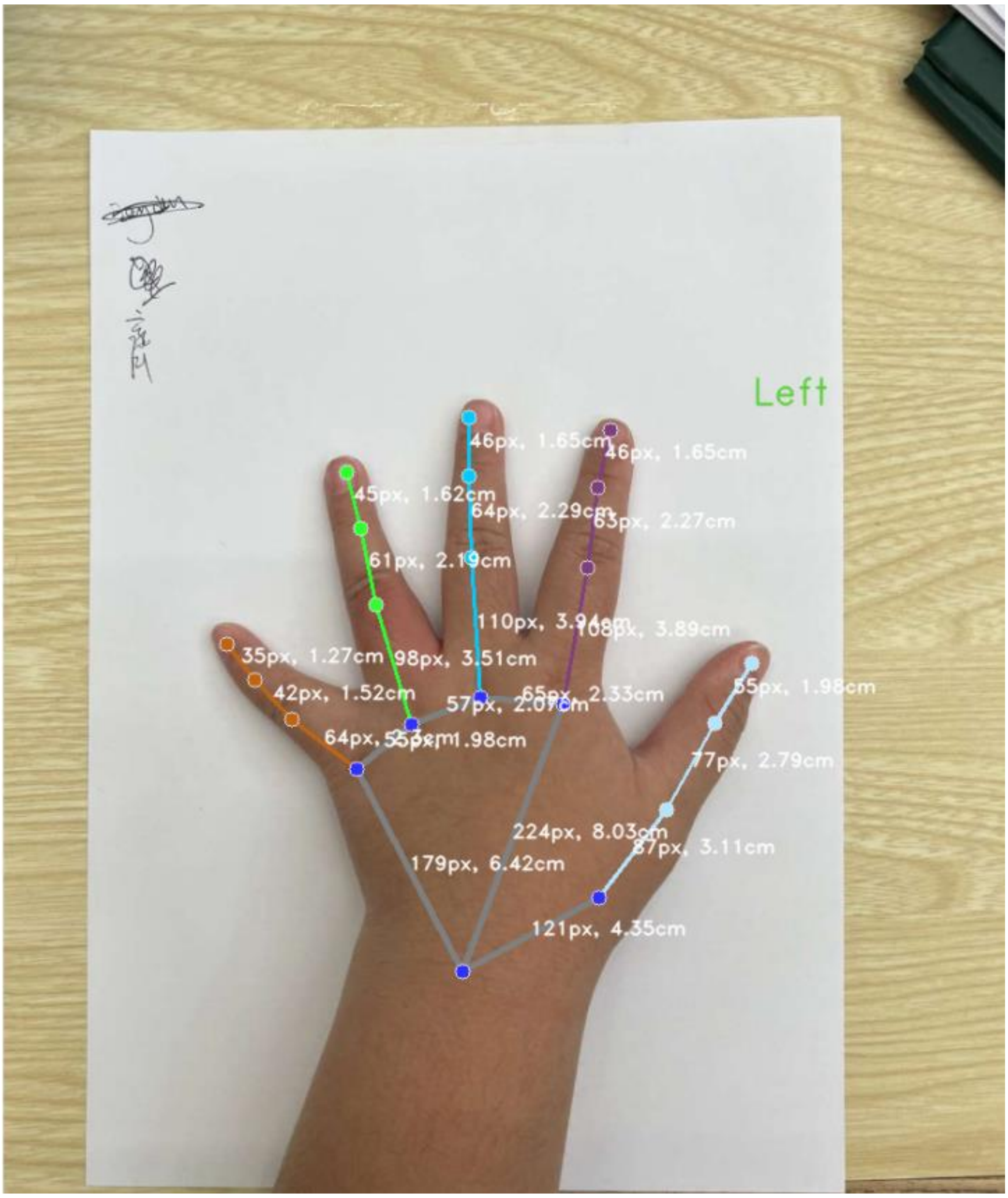
**Test Image 1:**




**Test Image 2:**

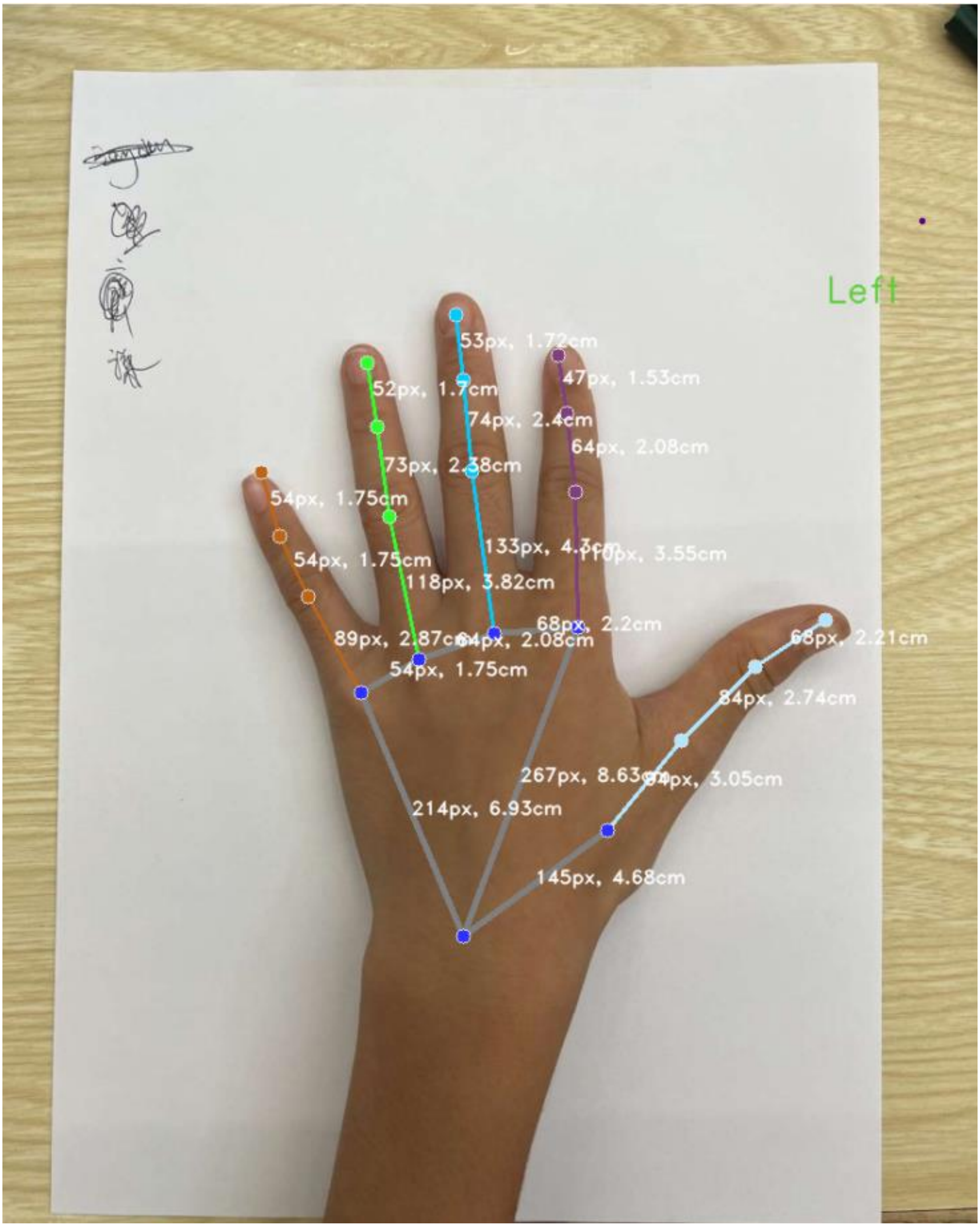
Sumedha Saxena
1155209203
BMEG5920

•



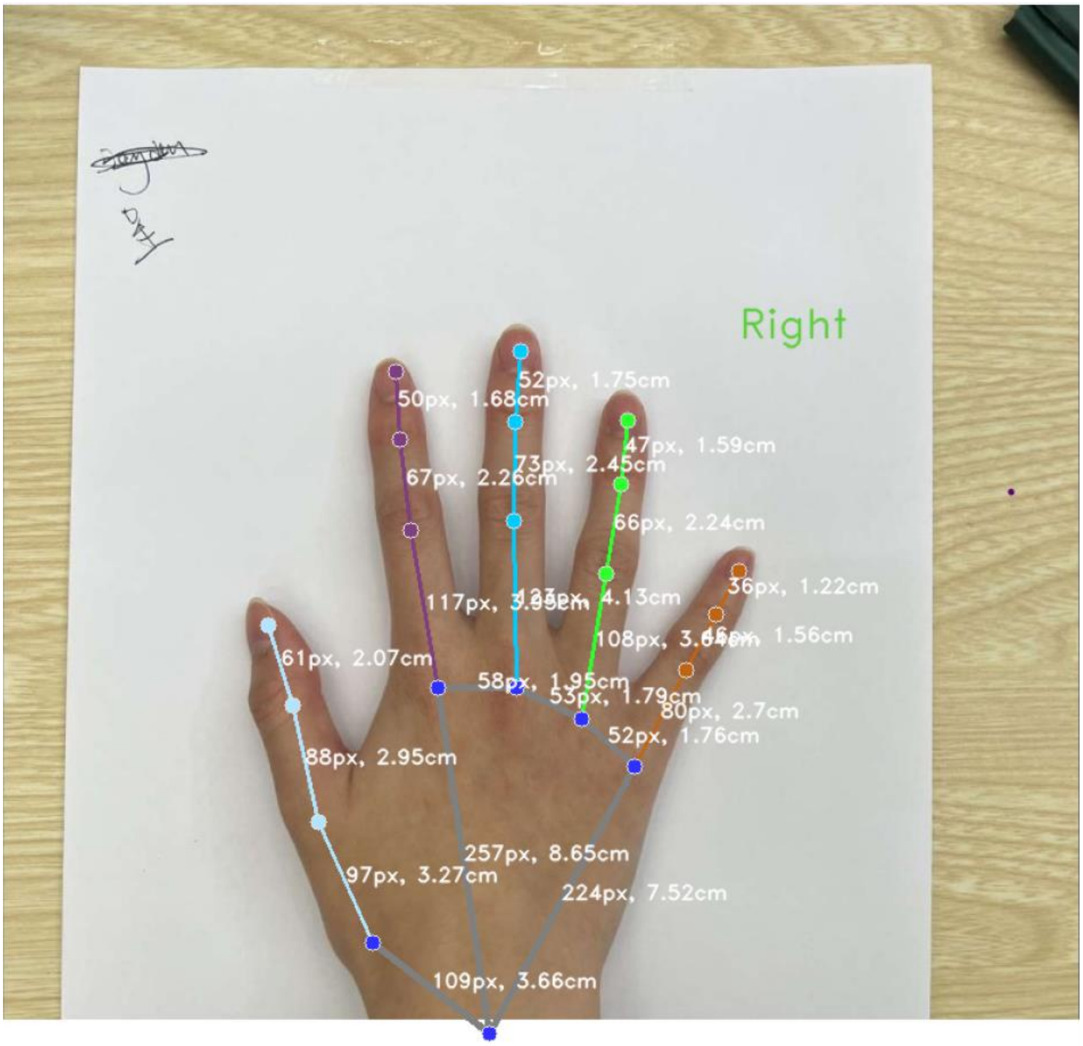

**Test Image 3:**

Sumedha Saxena
1155209203
BMEG5920




**Test Image 4:**

Sumedha Saxena
1155209203
BMEG5920




## 6. Improvements in current solution

So far, we have achieved a basic proof of concept for the original problem statement.

The results obtained for the test images need to be consolidated and checked against the actual measured distance to determine the accuracy and define an error tolerance.

The work that needs to be done in further refining the current implementation can be summarized as follows:

- Measure the accuracy of the detected distance against the actual distance for all test images.
- Improve the accuracy, if needed.
- Write a more robust logic to grab the correct coordinates of the A4 sheet corners for any input image.
- Code cleanup with some configuration options for easy handling of tasks.

## 7. Future work

For the upcoming semester, apart from the points mentioned above, we need to implement an end-to-end solution for Hand Landmark Detection.

This end-to-end solution can be a GUI Windows App, that should be able to let the user upload an image of the hand, run the steps detailed above and display/save the results in a way that can be passed on easily to the team which can utilize the info for stroke rehabilitation program.

I will be conducting research on the available GUI toolkits for implementation of the windows app and making the parameters as tuneable and configurable as possible.

## 8. References


Bo Sheng, J. Z. (2023). Commercial device-based hand rehabilitation systems for stroke patients: State of the art and future prospects. *ScienceDirect*. doi:https://doi.org/10.1016/j.heliyon.2023.e13588

Gupta, S. (2022). *python-libraries-for-machine-learning*. Retrieved from Springboard: https://www.springboard.com/blog/data-science/python-libraries-for-machine-learning/

*hand_landmarker*. (n.d.). Retrieved from MediaPipe Studio: https://mediapipe-studio.webapps.google.com/demo/hand_landmarker

*hand-keypoint-estimation*. (n.d.). Retrieved from MMPose: https://mmpose.readthedocs.io/en/latest/demos.html#hand-keypoint-estimation

https://www.sciencedirect.com/science/article/pii/S2405844023007958#bib5. (2023). *ScienceDirect*.

*Image Processing in OpenCV*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html

Liu, Y. H. (2018). Feature Extraction and Image Recognition with. *Journal of Physics*.

*MediaPIpe Solutions guide*. (n.d.). Retrieved from MediaPipe: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker


## 9. Current code location

The code for this project has been uploaded to GitHub at: [https://github.com/sumedhasaxena/HandLandmarkDetection/tree/master](https://github.com/sumedhasaxena/HandLandmarkDetection/tree/master)


Sumedha Saxena
1155209203
BMEG5920