Applying Deep Learning for Controlling Robotic Arm Gestures

Farhan Mahbub[1] and Md Sadatuzzaman Saagoto[1]

[1] World University of Bangladesh, Lake View Road, Uttara-17, Dhaka, Bangladesh

Abstract. The goal of this project is to control a robotic arm in real-time through computer vision. Live video feed is used as input for this project. The hand gesture is discovered in the video feed using Google's MediaPipe framework and sent to the Arduino Uno through serial communication. The servos of the robotic arms are programmed by an Arduino Uno in turn. This challenge required the use of MediaPipe and OpenCV. The code was implemented in Python. The present investigation investigates the possibilities for intuitive control of machines or robots through human-machine interaction. This project can address SDG 9 (Industry, Innovation and Infrastructure).

Keywords: Computer Vision, Hand Tracking, Python, MediaPipe, OpenCV, Arduino Uno

1 Introduction

1.1 Literature Review

In an era marked by rapid advancements in Artificial Intelligence and robotics, the intersection of deep learning and robotics has opened unprecedented possibilities for controlling and enhancing the capabilities of robotic systems. One fascinating application of this synergy is the utilization of deep learning techniques to control robotic arm gestures. Robotic arms are versatile tools employed in various industries, from manufacturing and healthcare and healthcare to space exploration and research. Traditionally, these arms have been programmed with precise and explicit instructions for their movements, limiting their adaptability and usability. However, by integrating deep learning algorithms, we can empower robotic arms with the ability to learn, adapt, and respond to complex and dynamic environments.

In the field of robotic control, the combination of MediaPipe, a strong framework for holistic perception, and OpenCV, a popular computer vision library, has gained popularity recently. The current state of research on the integration of OpenCV MediaPipe for robotic hand control is examined in this survey of the literature. Combining these technologies could lead to improvements in robotic manipulators' accuracy, versatility, and user interface. In an era marked by rapid advancements in Artificial Intelligence and robotics, the intersection of deep learning and robotics has opened unprecedented possibilities for controlling and enhancing the capabilities of robotic systems.

One fascinating application of this synergy is the utilization of deep learning techniques to control robotic arm gestures. Robotic arms are versatile tools employed in various industries, from manufacturing and healthcare and healthcare to space exploration and research. Traditionally, these arms have been programmed with precise and explicit instructions for their movements, limiting their adaptability and usability. However, by integrating deep learning algorithms, we can empower robotic arms with the ability to learn, adapt, and respond to complex and dynamic environments.

Wang et al work with the hand gesture at 2023,which is based on depth camera. In this research an assessment technique appropriate for this framework. In order to achieve effective object tracking and maximize the detection effect, a standardizer is also built. Inference time is ultimately shortened by 35%. This study architecture's implementation is available for public use at https://github.com/DumbZarro/BuddHand.[1] Wang et al also proposed suggestion is a quick FER algorithm that can track a driver's emotions and function well on low-end car electronics. In order to increase the accuracy of the classifier, a hierarchical weighted random forest (WRF) classifier is used, which is trained based on how similar the sample data is. The initial phase involves extracting geometric information from input photos by detecting facial landmarks and taking into account their spatial positions. The suggested hierarchical WRF classifier then uses these feature vectors to categorize face expressions.[2]

In the recent time, Aryan Gupta et al developed a touchless hand gesture detection arm. Using Google MediaPipe, a machine learning (ML) pipeline that combines palm detection and hand landmark models, we develop a simple hand tracking method to control a Robot Operating System (ROS) based surveillance car through socket programming. The investigation controls the direction and speed of a ROS car's steering. Surveillance vehicles that work using hand signals could improve security protocols.But Ahmad Puad Ismail et al used opencv with python in his work at 2021. The Region of Interest (ROI) theory along with Python programming will be used to recognize hands. Since the source code to read the real-time input video differs for the hardware implementation, the description of the results will concentrate on the simulation portion. The theories of hand segmentation and the hand detection system, which use the Haar-cascade classifier, can be used to the development of hand gesture recognition using Python and OpenCV.[3] James Davis et al developed a model-based technique for hand gesture recognition in humans. A generic gesture is modeled as four qualitatively different phases using a finite state machine. To calculate motion trajectories, fingertip tracking is done in numerous frames. These trajectories are then utilized to determine the gesture's start and stop positions. GesSures are represented as a list of vectors, and table lookup based on vector displacements is used to match them to gesture vector models that have been stored. Findings are displayed that demonstrate the identification of seven gestures with pictures captured at 4 ttz on a SPARC-1 without the need for specialized hardware. The left, right, up, down, grab, rotate, and stop actions are represented by the seven motions[4].Gurav et al do the exact thing of real time finger tracking in his work. Various machine learning techniques, including neural networks, support vector machines, and adaptive boosting (AdaBoost), constitute the foundation of hand gesture identification systems. To strengthen the detector, AdaBoost-based hand-pose detectors are trained using a smaller collection of Haar-like features.

For more than four hand movements, the matching context-free grammar-based suggested method provides strong and accurate real-time performance. Rectangles are causing some issues, therefore we have implemented the alternative representation approach (fingertip detection using the convex hull algorithm) for the same movements.[5]

There are also some work to control the robot hand using opencv & mediapipe From 2019 to 2023[6]–[10].We try to find out the lagging & gap from this research to solve here.

Broadly, tracking finger movements can be done in two ways. In the first method, one can use their own dataset, create a machine learning/ deep learning model, and train the model using the dataset. This approach requires a lot of work and is obviously a complex one. Another approach is to use a pre-trained model like MediaPipe or Handtrack.js and get the work done in a very simple and efficient way. In this project the later method is utilized. Between these two models, MediaPipe and Handtrack.js, MediaPipe is used. This is because Handtrack.js is a web browser-based model whereas MediaPipe is a more versatile, efficient, and easy to use model developed by Google.

2 Research Methodology

This is an experimental project, and all the programs are written in Python and C++ (written in Arduino IDE). The goal is to track fingers' positions in real time and control the robotic arm's finger accordingly. The project can be sub-divided into 4 parts:

1. Capturing and analyzing real-time video feed from the webcam.
2. Tracking the fingers' position (Open or Closed).
3. Simultaneous communication between Python and Arduino.
4. Controlling the servos of the robotic arm.

For the vision related tasks OpenCV is used. It captures the video from the webcam, converts the frames into images. After that the color profile of the images is changed to RGB from BGR.

2.1 Google's MediaPipe

For tracking hand movement and hand landmarks Google's MediaPipe framework is used. MediaPipe has hand tracking solution which provides a pre-trained model. This pre-trained model can be implemented without any complex programing for hand movement recognition. In this project this handy and user-friendly tool is utilized. MediaPipe's hand module has 21 landmarks for five fingers.

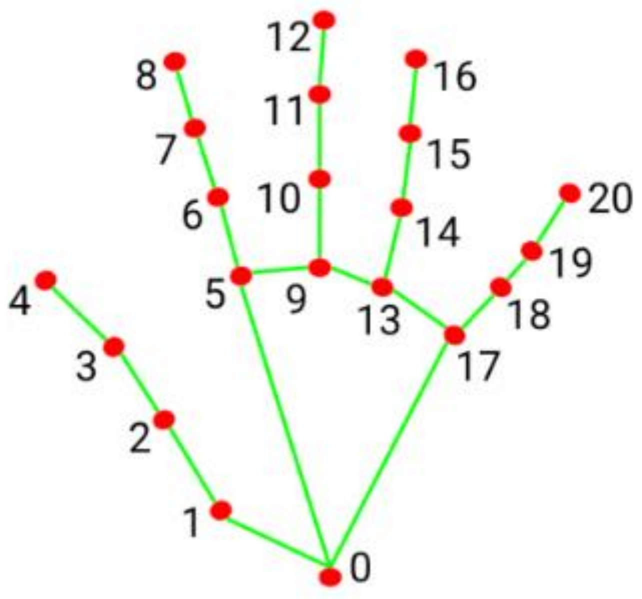
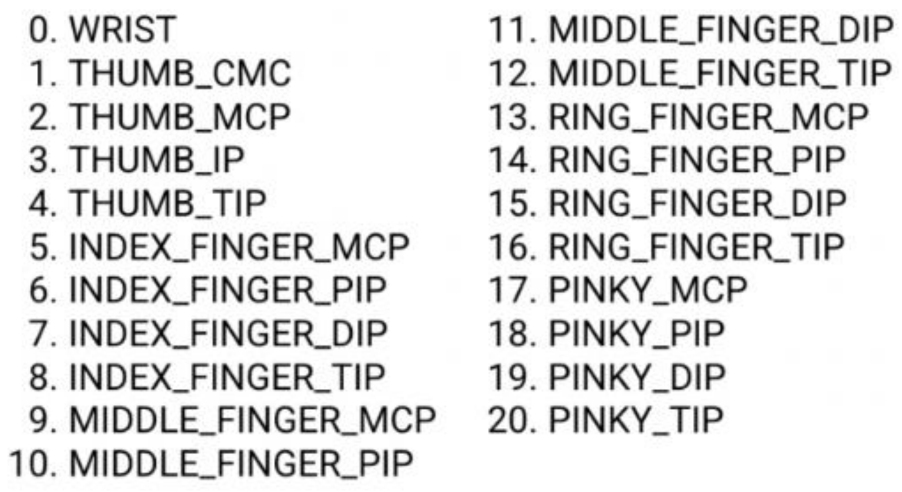


Fig. 1. Hand Landmark



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fig. 2. Definition of the 21 landmarks

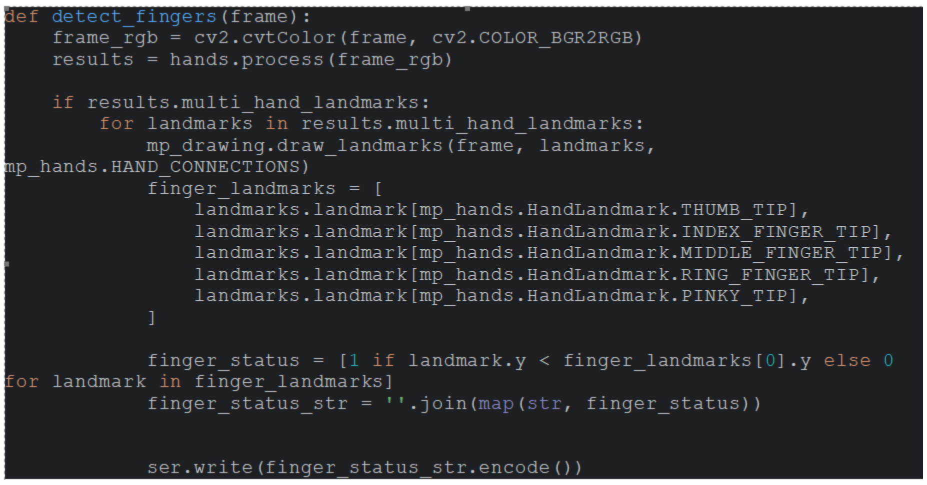
This hand landmarker model was trained on approximately 30K real-world images with several rendered synthetic hand models.

This model can recognize common hand gestures.

```
0 - Unrecognized gesture, label: Unknown
1 - Closed fist, label: Closed_Fist
2 - Open palm, label: Open_Palm
3 - Pointing up, label: Pointing_Up
4 - Thumbs down, label: Thumb_Down
5 - Thumbs up, label: Thumb_Up
6 - Victory, label: Victory
7 - Love, label: ILoveYou
```

2.2 The Program

The program detects whether a finger is raised or not using the tip of the thumb as reference. If the value of Y-coordinate of a fingertip landmark is higher than the value of thumb tip landmark than the program detects the finger as raised. The real-time input is taken from the webcam video feed.

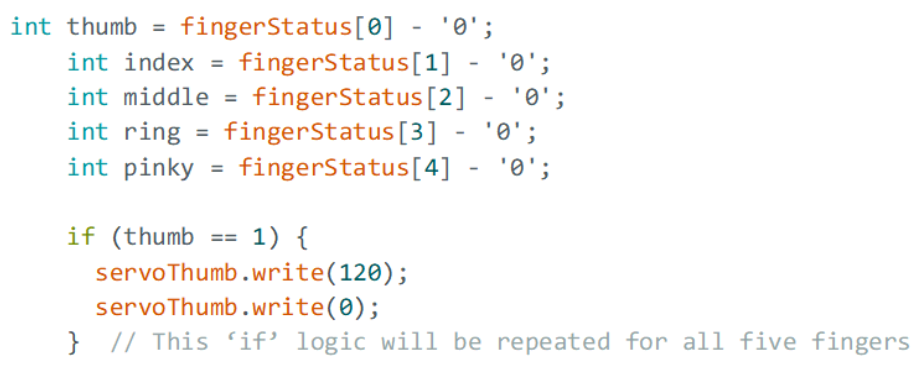


```
def detect_fingers(frame):
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        for landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks,
mp_hands.HAND_CONNECTIONS)
            finger_landmarks = [
                landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP],
                landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP],
                landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_TIP],
                landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_TIP],
                landmarks.landmark[mp_hands.HandLandmark.PINKY_TIP],
            ]

            finger_status = [1 if landmark.y < finger_landmarks[0].y else 0
for landmark in finger_landmarks]
            finger_status_str = ''.join(map(str, finger_status))


            ser.write(finger_status_str.encode())
```

Fig. 3. Python Code to Detect Fingers

2.3 Arduino Program

After detecting the finger's position, the code is sent to Arduino Uno via serial communication from Python. The Arduino code is then written which converts the data sent from Python into the language of the Arduino Uno microcontroller which in turn controls the servo.



```
int thumb = fingerStatus[0] - '0';
    int index = fingerStatus[1] - '0';
    int middle = fingerStatus[2] - '0';
    int ring = fingerStatus[3] - '0';
    int pinky = fingerStatus[4] - '0';

    if (thumb == 1) {
      servoThumb.write(120);
      servoThumb.write(0);
    }  // This ‘if’ logic will be repeated for all five fingers
```

Fig. 4. Arduino Code to Receive Data, convert them to Control Servos

2.4 Controlling Servos of the Robotic Arm

Python sends an array to the Arduino like [00000]. Here the 1st array position tells the position of the thumb, 2nd for the index finger and goes on till the pinky finger. The Arduino program converts this array to a binary 1 (finger raised) and 0 (finger closed) position. Next the command is sent to the servos to act accordingly.

3 Experimental Setup

For the project, the requirements are as follows:

3.1 Hardware:

1. Arduino Uno
2. Robotic Arm
3. Servo motor
4. Bug Converter
5. LiPo Battery (12 V)
6. Webcam
7. Laptop/Desktop for Computation

This hand tracking can be done without a wired connection. In that case, Bluetooth module or wifi module must be used.

3.2 Programming Language:

1. Python
2. C++ (Arduino Ide)

3.3 Library/Framework:

1. OpenCV
2. MediaPipe
3. PySerial

3.4 Editor/IDE:

1. Microsoft Visual Studio
2. Arduino IDE

In this project, volunteers were asked to participate. They were instructed to show their hand gestures in front of the webcam. Almost every time the hand movement and gestures were tracked correctly. It showed weakness in the case of detecting the thumb sometimes. Each time the code was calibrated.

4 Block Diagram

The block diagram of different connections of this project is

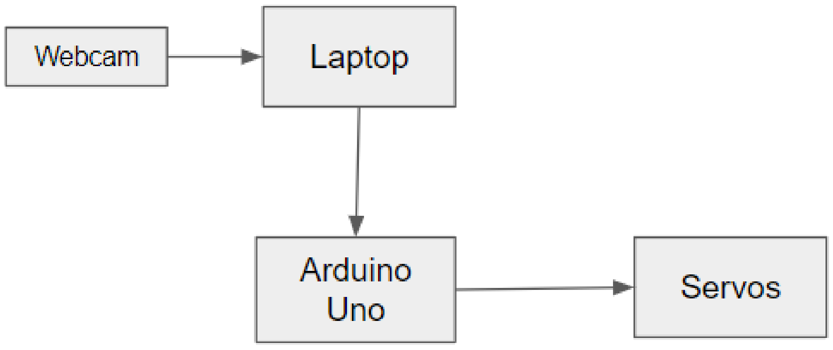


Fig. 5. Connections' Block Diagram

5 Results & Discussion

The program can detect finger position and movement in real time. The robotic arm is controlled accordingly. The use of MediaPipe makes the process faster and more efficient.

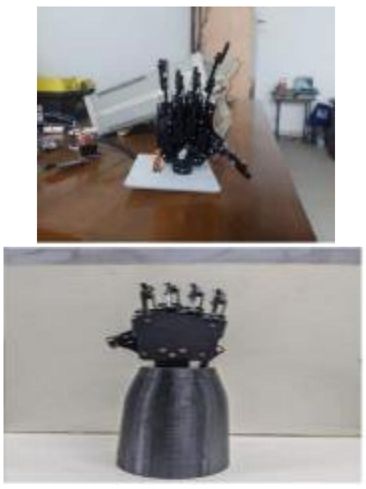


Fig. 6. Real-time Hand Tracking

MediaPipe framework provided the coordinates of hand landmarks in each frame of the video feed. All of these were done in real time. After finding the coordinates these commands were sent to the Arduino for servo control.
In this project, sometimes the thumb position is not detected correctly. In future work, this problem can be solved. Also, comparison with other methods and pre-trained model can be done in future.
This work may have applications in different fields. For instance, research labs which work with dangerous viruses/bacteria. Instead of humans interacting with these deadly molecules remotely controlled robotic arms can be used which reduces the health risks of the personnel.

This project did not require heavy computational resources. Any standard laptop configuration can do the work easily. As a result, the computation cost is very minimal of this project.

References

[1] Z. Deng, Y. Qiu, X. Xie, and Z. Lin, "A 3D hand pose estimation architecture based on depth camera," SPIE-Intl Soc Optical Eng, Mar. 2023, p. 18. doi: 10.1117/12.2671350.

[2] M. Wameed, A. M. ALKAMACHI, and E. Erçelebi, "Tracked Robot Control with Hand Gesture Based on MediaPipe," *Al-Khwarizmi Engineering Journal*, vol. 19, no. 3, pp. 56–71, Sep. 2023, doi: 10.22153/kej.2023.04.004.

[3] A. P. Ismail, F. A. A. Aziz, N. M. Kasim, and K. Daud, "Hand gesture recognition on python and opencv," *IOP Conf Ser Mater Sci Eng*, vol. 1045, no. 1, p. 012043, Feb. 2021, doi: 10.1088/1757-899x/1045/1/012043.

[4] J. Davis and M. Shah, "Recognizing Hand Gestures *."

[5] Institute of Electrical and Electronics Engineers Pune Section, IEEE International Conference on Industrial Instrumentation and Control 2015.05.28-30 Pune, and ICIC

2015.05.28-30 Pune, *International Conference on Industrial Instrumentation and Control (ICIC), 2015 28-30 May 2015, conference venue: College of Engineering Pune (COEP), Pune, Maharashtra, India*.

[6] M. Wameed and A. M. Alkamachi, "International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Hand Gestures Robotic Control Based on Computer Vision." [Online]. Available: www.ijisae.org

[7] Z. Deng, Y. Qiu, X. Xie, and Z. Lin, "A 3D hand pose estimation architecture based on depth camera," SPIE-Intl Soc Optical Eng, Mar. 2023, p. 18. doi: 10.1117/12.2671350.

[8] M. Jeong and B. C. Ko, "Driver's facial expression recognition in real-time for safe driving," *Sensors (Switzerland)*, vol. 18, no. 12, Dec. 2018, doi: 10.3390/s18124270.

[9] "4.IRJET_Gesture_Based_Wireless_Control_of".

[10] G. R. S. Murthy and R. S. Jadon, "A REVIEW OF VISION BASED HAND GESTURES RECOGNITION."

.