Gesture Imitating Robotic Arm

PROJECT REPORT

Submitted by

Vaishnav R

Register No: TVE16EC060

Sarath Vasu

Register No: TVE16EC044

Abhiram GP

Register No: TVE16ME003

to

The APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Electronics and Communication Engineering

Department of Electronics and Communication Engineering

College of Engineering Trivandrum

Kerala

July 2020

DECLARATION

I undersigned hereby declare that the project report, **Gesture Imitating Robotic arm**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Prof. Lalu V** and **Prof. Suresh Lal**. This submission represents our ideas in our own words and where ideas or words of others have been included, We have adequately and accurately cited and referenced the original sources. We also declare that We have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Thiruvananthapuram

Date: 15 July 2020

Vaishnav R

Sarath Vasu

Abhiram GP

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING TRIVANDRUM

CERTIFICATE

This is to certify that the Project entitled **"Gesture Imitating Robotic arm",** is a bonafide record of the project work done by **Vaishnav R**, **Abhiram GP** and **Sarath Vasu**, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Technology in Electronics and Communication Engineering from **APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY, KERALA.** This work is done during the academic year 2019-2020.

Prof. Lalu V
Assistant Professor
Department of ECE
(Guide)

Prof. Gigy PG
Assistant Professor
Department of ECE
(Project Co-ordinator)

Dr. Santhosh Kumar S
Department of ECE
(Head of Department)

ACKNOWLEDGEMENT

It is with extreme respect that we remember the names of all who had been a great help and guidance throughout our sessions. Firstly,We would like to thank the Almighty for giving me the wisdom and grace for presenting our project. With a profound sense of gratitude, We would like to express our heartfelt thanks to our guide, **Prof. Lalu V**, Assistant Professor, Department of Electronics and communication Engineering and **Prof. Suresh Lal S.R**, Assistant Professor, Department of Mechanical Engineering for their expert guidance, co-operation and immense encouragement in pursuing this project. We are extending my gratitude to our project coordinator, **Prof. Gigy PG**, Assistant Professor, Department of Electronics and communication Engineering for her co-operation and support. We are very much thankful to **Dr. Santhosh Kumar S**, Head of the Department of Electronics and communication Engineering, for providing necessary facilities and his sincere co-operation. Our sincere thanks is extended to all the teachers of the Department of Electronics and communication Engineering and to all our friends for their help and support.

Sarath Vasu

Vaishnav R

Abhiram GP

ABSTRACT

Robotics is the branch of technology that deals with design, construction, operation and application of robot. Nowadays most of the robot controls are using remotes, joy sticks, teach pendants, etc. Here we provide an alternative which eliminates the tool which the controller holds. Rather the controller himself becomes the input device.

We discuss configuration of a robotic arm which imitates the human gesture. The project consists of a mechanical arm which imitates the nearby human hand to produce the same motion. It uses concepts of video processing, 2-D pose estimation and servo control to achieve this task. A lot of papers have been scraped for new information regarding our project and come to a better understanding of the current scenario of human pose estimation. After trying and failing in using some of the techniques described in various articles and papers we came to the conclusion of viable alternatives to be used in our project. Different online platforms were used by us in completing our projects all of which are explained in this report. The firmware used by us to make our project a reality is also explained in detail.

This robotic arm can be used in wide ranging applications such as entertainment, industrial,healthcare and defence. They can replace human workers in harmful, potentially dangerous environments. It has various applications and even can serve military purposes. The wide ranging applications of this method is brought into view through our project which illustrates the opportunities of further development in this area.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machines are becoming an integral part of human life. Nowadays the machines has surpassed human abilities in many fields. Robots have taken over human jobs in a lot of areas and they are performing it better than ever. Figure 1.1 shows the major contributing fields of Robotics. Productivity has been hugely increased after the use of robots. And this can benefit workers and companies. When machines can replace the workforce, it gives workers time for intelligible works such as problem solving, idea pitching etc. Efficient controlling of these machines should be maintained. They could be controlled in many ways, they can be pre-programmed, controlled by commands or can even be autonomous. They perform tasks with much more precision and accuracy.

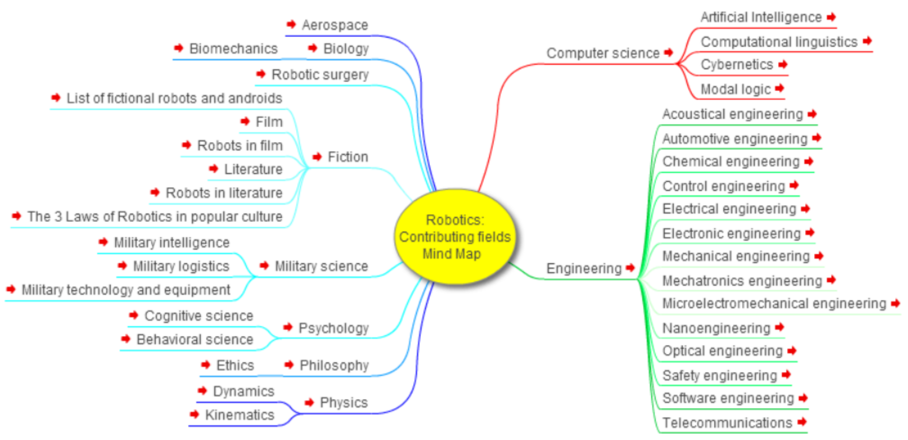

Figure 1.1: Major contributing fields of robotics

In most cases the robots are controlled by the written programs or a variety of feedback mechanisms. Technology is constantly improving its applications and the manner in which we communicate with the rest of the world. It is almost difficult to believe what the technology has accomplished. The advent of new technologies have provided us with better means to control robots. One of these is by the imitation of human movements by the machine. An advantage to this methodology is that the robot persistently surveys the motion while moving, considering real-time changes. The study of imitation in robotics has been motivated by both scientific interest within the learning process and practical desires to supply machines that are useful, flexible, and straight forward to use. The recent progress in Deep Learning has helped us visualize and predict human movements with high accuracy. The large data-sets available in the internet and various deep learning architectures has provided the machines with multiple ways to understand human motion and poses.

Gesture based recognition is one of the most encouraging areas of research. One of the earlier approaches to gesture imitating robotic arms is by the use of sensors like flex sensors or gyroscope to detect the human gestures. Here the gesture is found by the change in resistance on the flex sensor[3] or by the detecting the movement using gyroscope[2]. This approach is more costly and also the sensor becomes faulty after some use. Also the user has to wear a gloves in order to read the gestures. So this is not a suitable approach and a simpler and cost effective image processing based solution may be used. One such image processing technique is known as pose estimation. Pose estimation is a set of computer vision techniques that are used to find the pose of a human in images or videos, so that the action performed can be understood by the system.This is an effective technique in order to read the gestures.

The human pose estimation is among the rising concerns of the computer society as it is difficult to pinpoint specific points in space. Take a common scenario, as in the case of sports. In sprinting, its often troublesome to find the position of the participants if they are running close enough, causing a hindrance in determining the winner. This issue in localizing human joints via 2D imaging has heavy applications in ordinary life as well as in gaming and animation.

The basic idea of our project revolves around these concepts. It is a robotic arm that imitates the arm movements of a human. The robotic arm perceives the human motion using a 2D camera and then processes the incoming video to detect the human pose. The pose is a set of keypoints (bodyparts such as elbow,wrist etc.) and the corresponding confidence (estimation of keypoint, lies between 0.0 and 1.0).Then the robotic arm imitates the detected pose by using parameters that are found by manipulating the keypoints. By this, the movement of the robot can be controlled by human gestures which can be considered as one of the efficient ways of robot controlling methodologies.It serves a wide range of applications. This robot can be mainly used in situations which are dangerous or not suitable for a human like dealing with explosives,radioactive wastes or dangerous chemicals with higher precision and accuracy, thus providing a good and safe solution.

Chapter 2

Literature Survey

During the start of our project we had a clear aim for the final product of our work. We wanted a robotic arm which imitated the human arm movements. Our first goal was to make this work with a servo for the elbow position of the arm. Multiple methods with more than one cameras or IR sensors to read the depth was available at the time. Using Kinect sensor was one of the top options. But we were aiming for a more accessible solution. We wanted to use just a 2D video input from a normal camera. The popularity of Deep Learning techniques for estimating human poses from 2D images seemed like a good alternative to us.

We started our search for a good deep learning technique for human pose estimation. There were multiple trained deep learning architectures that estimated human poses. Each used different methods for training and different ideologies. This gave rise to different options each having its own merits and demerits.

A blog by Sudharshan Chandra Babu in nanonets.com named *"A 2019 guide to 3D Human Pose Estimation"* [1] helped us understand that 3D poses can be derived from 2D images. It can be done using 3 different ways as suggested by this blog.

1. Pictorial structure model (PSM) used in 2D pose estimation can be extended for 3D pose estimation.

2. Discriminative methods which view pose estimation as a regression problem.

3. Deep Learning Methods.

Finding that 3D pose estimation using 2D images is not viable for our project and also its much harder than other techniques we resorted to searching for other methods.

"Convolutional Pose Machines" [4] provided a sequential prediction framework for learning rich implicit spatial models. It seemed like a good method but the search for something better continued.

A more wide view on the topic was achieved on reading *"Deep Learning Based 2D Human Pose Estimation: A Survey"* [5]. It provided an eagle eye perspective of the topic and covered many basic concepts of human pose estimation. This article helped us achieve a better understanding of all the different methods and their uniqueness.

A some what different approach to the problem was found in the paper *"Mask R-CNN"*[6]. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. This method could be extended to mask humans and thus determine poses. But it was too much of an unnecessary challenge to go that way.

"Towards Accurate Multi-person Pose Estimation in the Wild"[7] was also visited during our searches. This also uses FasterRCNN in its first stage and then in the second stage the keypoints are detected inside the bounding boxes for the object.

"PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model"[8] tackles both semantic-level reasoning and object-part associations using part-based modeling. It employs a convolutional network which learns to detect individual keypoints and predict their relative displacements, allowing us to group keypoints into person pose instances. Further, it proposes a part-induced geometric embedding descriptor which allows us to associate semantic person pixels with their corresponding per-

son instance, delivering instance-level person segmentations. This model seemed promising and by studying more about this model we stumbled across *“OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”* [9]. In this work, a realtime approach to detect the 2D pose of multiple people in an image is presented. The proposed method uses a nonparametric representation, which is referred to as Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. This bottom-up system achieves high accuracy and realtime performance, regardless of the number of people in the image. This was actually a better version of the PersonLab model. We finally found us a point to start our project. We decided to use the openpose model for human pose estimation in our project.

Chapter 3

Methodology

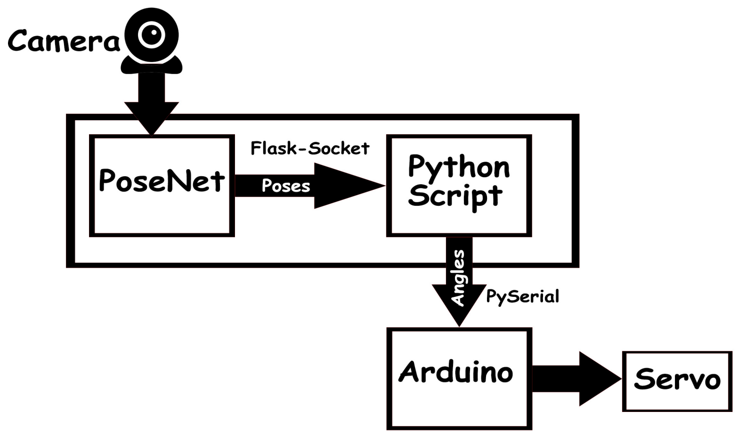

Figure 3.1: Overall framework of the project

3.1 PoseNet

The openpose provided the most accurate way to estimate human poses. As we started working on a basic pose estimation setup using openpose we found that we had to use GPU programming with CUDA and needed high end computer systems to get real-time feedback. But we wanted a more accessible and light weight option. Then we learned about posenet.

PoseNet is built to run on lightweight devices such as the browser or mobile device where as OpenPose is much more accurate and meant to be ran on GPU powered systems. These models used a bottom-up approach unlike most of the other models. They were trained on images to detect human keypoints which are basically joints like elbow, knee, wrist etc. Most of the earlier models used object detection to first detect a human and then find the keypoints. But the openpose paper proved that detecting the keypoints first and then joining them using Part Affinity Fields (PAFs) provided more accurate and faster results. The Figure 3.2 shows the keypoints detected.

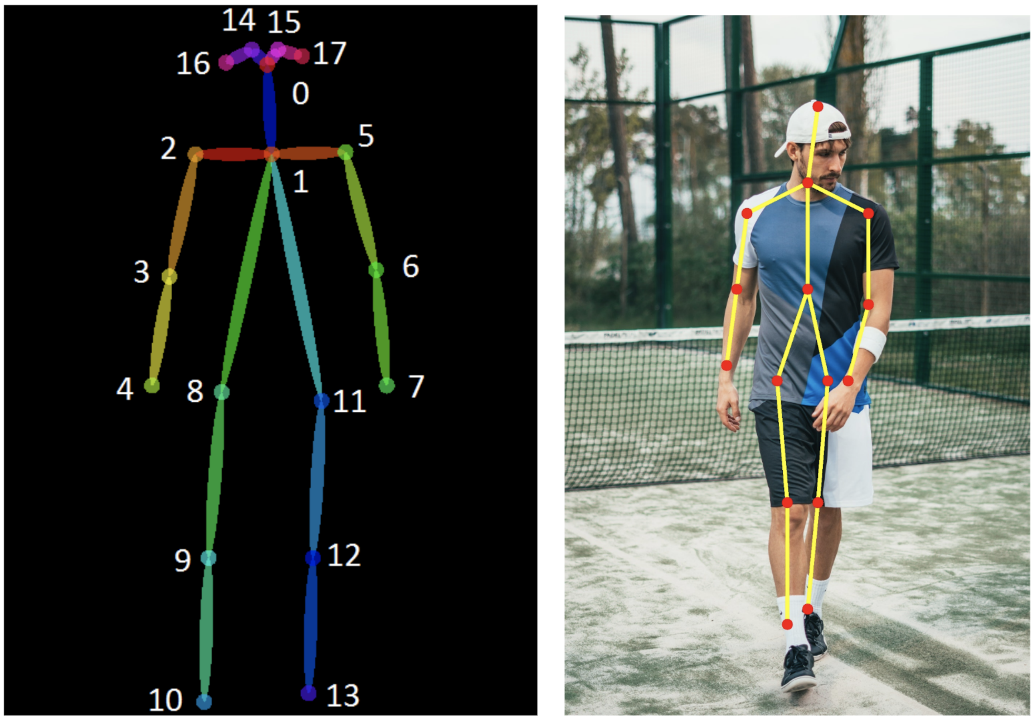

Figure 3.2: The Keypoints detected for pose estimation

Google released a Tensorflow.js version of Posenet model in 2018. It allowed real time human pose estimation in the browser itself. We could even tweak around a little bit and use a even lighter version of posenet that runs on mobiles.

PoseNet can be used to estimate either a single pose or multiple poses, meaning there is a version of the algorithm that can detect only one person in an image/video and one version that can detect multiple persons in an image/video. The single person pose detector is faster and simpler but requires only one subject present in the image. We have decided to use just the single person detector for better accuracy.

3.2 ML5.JS

Machine learning is getting more and more popular across the world.The Ml5.js is a library that enables you to get started with machine learning without many prerequisites. The ml5.js is a platform which makes machine learning approachable for a broad audience of artists, creative coders, and students.The Primary objective of this library is to make Machine learning user-friendly and familiar to everyone. This library provides access to machine learning algorithms ,task and models in the browser. Ml5.js is built over TensorFlow.js . So it uses the functionalities of TensorFlow.js at the backend, but it is made easy for people who are new to the Machine learning field. It consist of many machine learning models such as PoseNet, BodyPix, FaceApi etc . Ml5.js is a high level library, which means all models in this library are pre trained. Anyone with basic understanding of JavaScript and HTML can work with these models , so it makes this platform exceptional. This library provides immediate access to these models in the browser, and these can be utilised for human pose detection, composing music, image identification, generating text and much more.

It is an open source project developed and maintained by NYU's Interactive Telecommunications/Interactive Media Arts program and by artists, designers, students, technologists, and developers from all over the world. The browser based approach is the most important attribute of Ml5.JS . So it can be used in the browser itself without installing complicated dependencies. This platform provides us with a pre-trained PoseNet model which is used by us in our project. This platform simplifies the use of Posenet and has built in parameters, syntax, methods, event listeners etc.

3.3 P5.JS

The p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, and beginners. It is free and open-source platform. We built our JavaScript sketch on this online platform.

The ml5.js posenet library was imported into our sketch in p5.js and we started working on human pose estimation on the video input taken from our computers webcam. The posenet model is loaded and is connected to the video from our webcam in this javascript sketch we wrote. This allows our computer to get information from the cloud server that runs the posenet model. The thing we should understand as we go on is that the video that we collect from our webcam is sent to a cloud server frame by frame that runs the posenet model which then processes each of them and an output of 17 keypoints for each frame is given back to us with a probability value for each of them. Multiple poses with different probabilities and with each pose having multiple key-points of different probabilities are outputted. We get the position values of the pixels for each keypoints also as output. We use the pose with the highest probability of being true

3.4 DETECTING AND DRAWING THE HUMAN POSE

The first phase our project was to use our computers webcam to capture a video of a person and a real-time pose of the person is drawn on the computer screen as the person moves. For this as we explained in the section 3.3 we used the posenet library from ml5.js to write a javascript code in p5.js. Each frame sent to the cloud server which runs posenet returned an array of poses. We took the first object of that array i.e, poses[0] and for each keypoints position we created variables to store the dynamic values. There were 34 variables which stored the values of positions x and y for all the 17 keypoints. The 17 keypoints are shown in Table 3.1 .

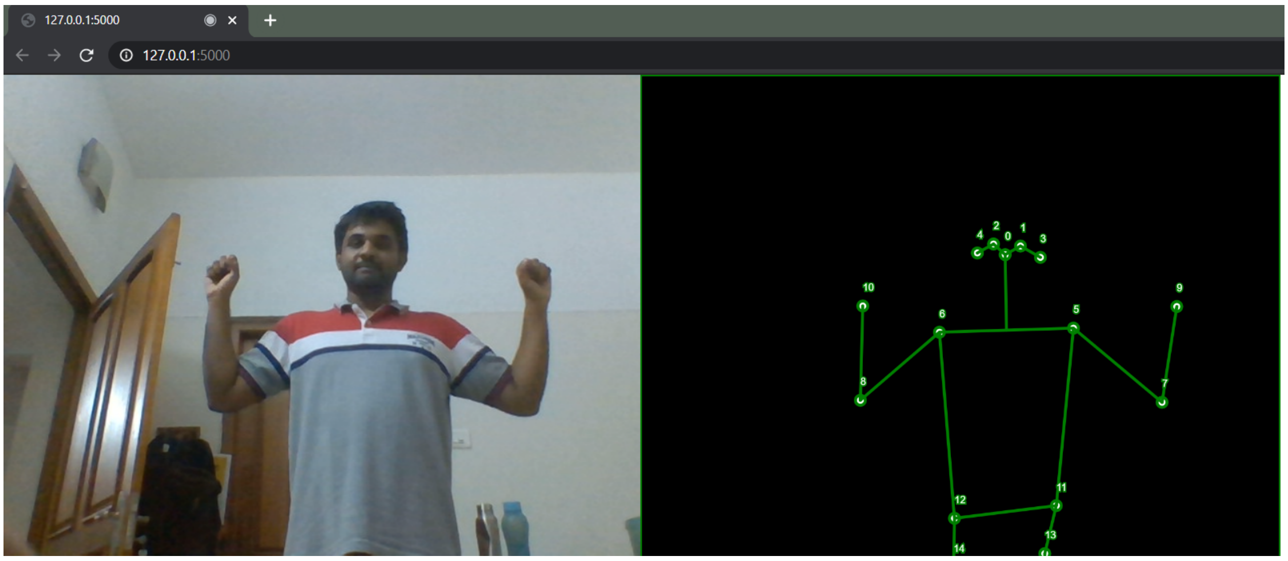

Figure 3.3: The Keypoints estimated shown on browser

Since these variables kept changing a lerp function was used to smoothen the shift of the keypoints. Now these keypoints were joined together and lines were drawn over the video to produce a pose skeleton. We wrote our code such that as the webcam records the video of a person two screens are shown in the computer.This is shown in Figure 3.3 One screen shows the original video and the second screen shows the estimated pose of the person drawn using green lines with highlighted dots for each keypoint.

Table 3.1: Keypoints with respective id

id	part	id	part
0	nose	9	leftWrist
1	leftEye	10	rightWrist
2	rightEye	11	leftHip
3	leftEar	12	rightHip
4	rightEar	13	leftKnee
5	leftShoulder	14	rightKnee
6	rightShoulder	15	leftAnkle
7	leftElbow	16	rightAnkle
8	rightElbow		

3.5 Connecting the Javascript with Python code using Flask-SocketIO

Now that we have the code for realtime pose estimation the next thing was to calculate the angles of movementfor the right arm. Here we are just concentrating on imitating the right arm movement of a person. So we thought it would be better to sent these pose values to a python code where we could process the values to determine the angles and then use the output to control the relevant servo mechanisms.

Flask-SocketIO [10] gives Flask applications access to low latency bi-directional communications between the clients and the server. The client-side application can use any of the SocketIO official clients libraries in Javascript, C++, Java and Swift, or any compatible client to establish a permanent connection to the server. Flask-SocketIO is compatible with both Python 2.7 and Python3.3+. When using this SocketIO, messages are available to both parties as events. Flask-SocketIO also supports SocketIO namespaces, which allow the client to implement several independent connections on the same physical socket. After installing Flask package, we have imported the Flask socket libraries to our python script. We have added an html script in the "index.html" file inorder to load the Socket.IO library and establish the connection. Using this, we established a socket between our javascript code and a python script named “main.py”. And the poses are sent to the python script in real-time. This data is used for future angle detection algorithm.

3.6 Calculation of Angles between joints

The values of pose from the array of poses sent from the browser reached the python script in JSON format. From here we only made variables to store the values for the right shoulder, right elbow and right wrist so that we could calculate the angle at the elbow. A python function was defined which takes in values of the x and y positions of three points and outputs the value of angle between them.

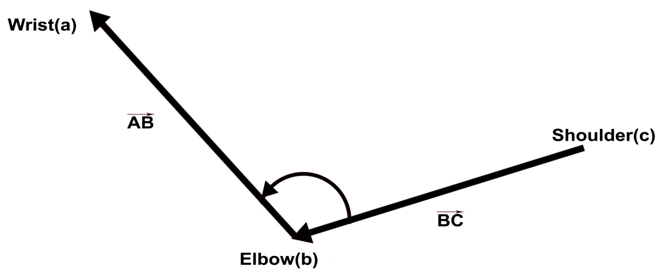

Figure 3.4: Angle between the joints

let us consider 3 points a,b and c as wrist,elbow and shoulder respectively as in Figure 3.4 .Now inorder to find angle between them ,lets find the vector $\vec{AB}$ and $\vec{BC}$ as follows

$$\vec{AB} = b - a$$

$$\vec{BC} = c - b$$

The Angle between the joints (θ) is found by computing dot product between the 2 vectors.the angle θ is found as

$$\theta = \arccos \frac{\vec{AB}.\vec{BC}}{|\vec{AB}||\vec{BC}|}$$

3.7 Connecting the Python code to Arduino

Using the Py Serial [11] library in python we established a serial communication with an arduino. The arduino was the microcontroller used to control a servo which controlled the elbow movements of our robotic arm. The port to which arduino is connected and the baud rate is initialised first. Then we sent the angle values through the serial communication port to the arduino which in-turn sent signals to the servo motor. The angle calculated is typecasted to a string. This is because we could only send data as bits to the serial port. Thus character bits are sent one by one to the serial port.

If we need to send 2 values of angles that is to control the movement of 2 joints then we need to append one of the angles to the other separated by a space in between. The angles could then be separated at the arduino.

3.8 ARDUINO SKETCH

Arduino sketch uses the serial library for the serial communication and the servo library for controlling the servo. First the baud rate was set and the serial communication is established.We set a baud rate of 9600 this was the exact same baudrate that we set in the python script. Now we check if any data is available in the serial port. If data is present it reads the string until a new line comes up. The string is then converted to an integer data type and is used to control the servo. The servo has to be connected to a PWM pin of the arduino and has to be initialised first using the Servo.attach( ) function. The servo is then adjusted to its initial position and after data is received the servo moves to the position needed.

But when we tested the algorithm we found a small problem that needed to be rectified. The problem was that the python code was sending real-time continuously variable angle values to the arduino and the servo didn't get enough time to adjust its position between incoming angle values. So we decided to send the angle values with a time gap between each. To introduce this time gap we went back to the JavaScript code and adjusted the time gap between the pose values that was sent to the python code through the socket. For this no kind of time delay function was introduced we used a different method. A variable i was introduced with a value of 0. After each pose is drawn in the screen by the JavaScript code a value of one was increased to the variable i. After a certain value of i was reached it was reset to zero value and the pose values at that instant was sent through the socket to the python code. So this introduced a time delay between poses reaching the python code and thereby angle values reaching the arduino. Experimentally through trial and error we found that 6 was the correct threshold value for the variable i. Each time i hit the value 6 a pose was emitted through the socket and simultaneously the value of i reset to zero. Now we had a smoothly functioning servo which sets to angles as described by the right elbow angles of a person standing in front of the webcam.

If we need to control 2 joints then we require 2 angles. The angles are appended and sent as a single string. There is a function in the string library known as strtkn_r( ) to separate a string at a delimiter. If we use this function and use “ “ (space) as the delimiter then we can split the 2 strings and typecast it to an integer data. but inorder to detect the delimiter in the string, the string needs to be converted to a character array using the toCharArray() function. Thus we can control multiple servos with this method. But as the number of servos increases the splitting process takes more time and hence we may experience a delay between the pose estimation and the gesture imitation.

Chapter 4

Firmware

The hardware part in the project includes a microcontroller and actuator. Initially we were planning to use a raspberry pi as the microcontroller as it has better processing power ,but due to the corona virus pandemic we were not able to procure it thus we had to move to an arduino which was available to us. Also we used servo motors as the actuators.

4.1 ARDUINO

We used an Arduino UNO for this project. It is a microcontroller board based on ATmega328P. It has 14 digital pins, 6 analog pins, a USB connection, a power jack, 16 MHz crystal resonator and a ICSP header. The ATmega328P is a 8-bit AVR RISC-based microcontroller which combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM and 2KB SRAM. Thus,the ATmega328P has a low computing power and thus could only be used to actuate the motors. It cannot be used to perform high calculations and thus we process the information in the computer and send it to the arduino using serial communication. Figure 4.1 shows an Arduino uno.

Figure 4.1: Arduino UNO

We preferred raspberry pi as it is basically like a computer and can run the python script, JavaScript and also support a webcam. Thus we could have avoided any connection to the computer and make the entire project more mobile. Now in this case, we use the integrated camera of the computer to input the frames. The frames are processed using the posenet model to get the Pose and they are given to the python script through the flask-socket to get the angles. The angles are sent to the serial port in which arduino is connected using the py-serial library. The arduino sketch uses the Serial.Read() function to read the values from the serial port and are then used to actuate the motors.

We use the arduino IDE is used to write and upload the code to arduino. it is an open-source platform available. It can be cofigured for any of the arduino boards available like Arduino UNO, Nano, ESP32, ESP8266 etc.

4.2 CAMERA

As Arduino lacks the processing power,we use the computer itself to process the frames and find the poses. The camera access is chosen in the browser and is an integral part of the project. The camera can be an integrated camera or an external camera. We used the integrated camera in the computer which is a 5 Mega Pixel HD camera.

4.3 SERVO

Servos are rotary actuators that can be controlled with great precision. It can be rotated at specific angles or distance and thus is a very useful for making precision movements. Servos are small in size but provide high torque and are efficient. The servos are mainly used for industrial applications, robotics and in line manufacturing. The servo circuitry is placed besides the motor unit and a positionable shaft with gears are also present as shown in Figure 4.2. An electric signal is used to control the movement of the servo.

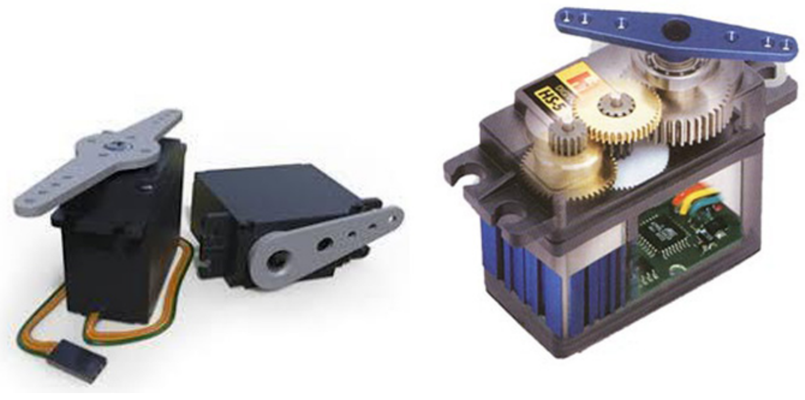

Figure 4.2: Servo motor

The servo is controlled by an electrical pulse of variable width i.e., pulse width modulation (PWM). The servo can turn 90° in either direction for a total of 180°. The neutral position of the servo is where it could move equal distane in both the clockwise and anticlockwise directions. The pulse sent to the motor determines its position of the shaft and the pulse width determines the angle the motor has to move. The Figure 4.3 shows the pulse width required for the servo to move 0°, 90° and 180°.

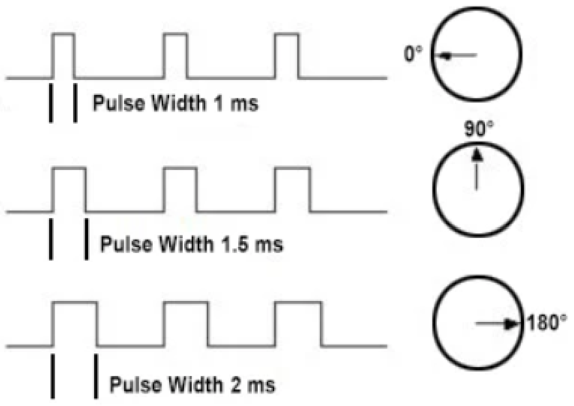

Figure 4.3: Pulse width for 0°, 90° and 180°

The arduino has a servo library in order to control it. It contains many function to control the servo such as moving to a specific angle and to read the current position. The servos are connected to the PWM pins of the arduino. In the case of our project, the angle to be moved is sent to arduino from python and the servo moves to that position in order to mimic the gesture.

Chapter 5

Future Scope

Our project can only mimic gestures in a 2D plane and not all gestures could be imitated. Thus the algorithm we prepared can only be used to control a robotic arm with 3 Degree of Freedom (D.O.F). This could be improved to a robot arm with 4 Degree of freedom if we incorporate the movements in a 3D perspective. If we consider that the image is a projection of a 3D object on a 2D plane, then with this we could find the angles. For this we need to calibrate the length of the arm in the image at a particular distance from the camera and then use it for the calculation of some more angles that could give us a bit more movement. So by using a distance sensor to calculate distance between camera and the person we could scale the initial calibration to the required value and thus could get a more accurate imitation.

After the release of Raspberry Pi4, which has more processing power than its predecessors the control would be more smooth and can provide a more robust solution. Also if we use high torque servos or more powerful motors, then it could be used for many practical situations.

With the ever increasing technology, the processing power of micro-controllers are increasing at a rapid rate, in the near future it may be even possible to control a humanoid robot completely with this technology. also with the improvement of the Virtual reality it may be possible to control a humanoid remotely as if we are present there.

Also there is a version of posenet that could be run on the mobile phones using the Mob net architecture. Using this feature we can create a mobile application in order to control the robotic arm. Now a days as the mobiles are getting better processors and cameras, thus it would be more accurate and simpler solution. Thus there are further improvements that could be made to make it better to use.

Chapter 6

Conclusion

Our robotic arm imitates human gestures by detecting the human pose using PoseNet model. Posenet is a machine learning model used for human pose estimation in the browser using tensorflow.js. Using posenet and JavaScript ,we get the keypoints.These are sent to python using flask socket. From these keypoints, we successfully created an algorithm for detecting angle between the joints. The angles were sent from computer to arduino using the serial port. We tested the algorithm on a model arm we built using arduino and mini servos. We were planning on fabricating the arm using high torque servos and raspberry pi, but were not able to procure the components due to the corona virus pandemic.

The entire process creates a small delay but it is small enough to be ignored and doesn't disturb the entire process. This is due to the arduino board , where the communication between arduino and computer takes place serially. But this delay could be reduced by using a micro controller, in which the entire code could be processed. The keypoint detection is becoming more faster and accurate, also new keypoints other than the 17 keypoints available are being introduced such as for the fingers. But currently the finger keypoints takes a lot of time to process and real time detection is still a far away concept. Thus if the finger keypoints also are used we could completely imitate the gestures of a human hand.

This robotic arm can be used in wide ranging applications such as entertainment, industrial, healthcare and defence. They can replace human workers in harmful, potentially dangerous environments such as mining. The elderly can use the arm to perform daily tasks without having to get up or move about. They can simply sit at one location and be able to control the arm through gestures. In the medical industry mechanical arms may be used to perform complex surgeries in the absence of availability of a doctor or in case of an extreme emergency. in the future a humanoid robot could be made and controlled using the gesture imitation. Then it would be possible to explore and analyse places where human beings can't survive.

References

[1] **"A 2019 guide to 3D Human Pose Estimation"** , *https://nanonets.com/blog/human-pose-estimation-3d-guide/*

[2] **"Wireless Mobile Robotic Arm"** , *Mohd Ashiq Kamaril Yusoffa, Reza Ezuan Saminb, Babul Salam Kader Ibrahimc, International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012), July 2012.*

[3] **"Robotic ArmControl through Human ArmMovement using Accelerometers"** , *Ashutosh Pattnaik, Rajiv Ranjan,*

[4] **"Convolutional Pose Machines"** , *by Shih-En Wei, Varun Ramakrishna, Takeo Kanade and Yaser Sheikh*

[5] **"Deep Learning Based 2D Human Pose Estimation: A Survey"**,*by Qi Dang, Jianqin Yin , Bin Wang, and Wenqing Zheng*

[6] **"Mask R-CNN"**, *by Kaiming He, Georgia Gkioxari, Piotr Dollar and Ross Girshick from Facebook AI Research (FAIR).*

[7] **"Towards Accurate Multi-person Pose Estimation in the Wild"** , *by George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, and Chris Bregle*

[8] **"PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model"**, *by George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, Kevin Murphy*https://flask-socketio.readthedocs.io/en/latest/

[9] **“OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”** *,by Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE, Tomas Simon, Shih-En Wei, and Yaser Sheikh*

[10] **Flask-SocketIO**,*https://flask-socketio.readthedocs.io/en/latest/*

[11] **pySerial**,*https://pythonhosted.org/pyserial/*