**HEAD-POSE CONTROLLED ROBOT**

*A Graduate Project Report submitted to Manipal Academy of Higher Education in partial fulfilment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**Electronics and Communication Engineering**

*Submitted by*

**Sai Tarun Madhira**

Reg. No: 150907046

*Under the guidance of*

**Dr. Nakul Shetty & Dr. Ramya S. Lokesh**

**Associate Professor Associate Professor**

**MIT, Manipal MIT, Manipal**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**MANIPAL INSTITUTE OF TECHNOLOGY**

(A Constituent Institution of Manipal Academy of Higher Education)

MANIPAL – 576104, KARNATAKA, INDIA

**MAY/JUNE 2019**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**MANIPAL INSTITUTE OF TECHNOLOGY**

(A Constituent Institution of Manipal Academy of Higher Education)

MANIPAL – 576 104 (KARNATAKA), INDIA

Manipal

18/09/2019

**CERTIFICATE**

This is to certify that the project titled **HEAD-POSE CONTROLLED ROB** is a record of the bonafide work done by **SAI TARUN MADHIRA** (*Reg. No. 150907046*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (BTech) in **ELECTRONICS AND COMMUNICATION ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institution of Manipal Academy of Higher Education), during the academic year 2018 - 2019.

|  |  |
| --- | --- |
| **Dr. Ramya S. Lokesh**  *Associate Professor, ECE*  *M.I.T, MANIPAL* | **Prof. Dr. M. Sathish Kumar**  *HOD, ECE*  *M.I.T, MANIPAL* |

**ACKNOWLEDGMENTS**

The completion of my final semester project would not have been possible without the help and guidance of my respected guides. I express our gratitude to all the mentors mentioned below. Dr Ramya S. Lokesh, my internal guide, mentor to me from the Department of Electronics & Communication, Manipal Institute of Technology, Manipal. She has helped me through the project by ensuring that i have understood all the concepts to have a strong base of fundamentals. She has shown me the importance of diligence and also stressed on the importance of theoretical as well as practical work. She has helped me contact academic experts, to gain their advice in the project’s course. I thank Professor Dr. Ramya S. Lokesh, for constantly supporting us during the course of our project and thank her profoundly for the spirited guidance and continued encouragement. Dr. Nakul Shetty K. my internal guides, allocated to me from my branch, Department of Electronics and Communication. Both the respected guides provided me with the prerequisite knowledge for the smooth running of the project. Timely passing of information, well in advance to keep me updated with the project deadlines and encouraging me to push my limits further by enriching my knowledge in the field. They provided me with valuable instructions that helped us to ensure that this project has derived a detailed and reliable analysis, keeping on track with the project schedule and maintaining punctuality. I extend my sincere gratitude towards Dr. Nakul Shetty K, professor of Department of Electronics and Communication Engineering. I would also like to thank the faculty and staff of Department of Electronic and Communication Engineering, without whose constant support and assistance, this project would not have had a smooth run. I would also like to extend our regards to Professor Dr. M. Satish Kumar who gave me valuable insights into my project and helped me move in the right direction and helped sort out any administrative issues that came up. Lastly, I would like to thank our institute, Manipal Institute of Technology and the University, Manipal Academy of Higher Education for equipping me with the tools required to complete this project and giving me an opportunity to be a part of such a great experience. They have helped me grow as an individual and also have given me a lot of valuable knowledge.

**ABSTRACT**

Gesture technology has becoming increasingly common over the past decade. Gesture controlled games, driver safety systems, wheelchairs, drones etc. have started becoming a popular presence in the market. With development rapidly growing in this field, people are trying to integrate it more and more with common devices to allow a more natural and functional means of human computer interaction (HCI). They allow for hands free control, thereby leaving them to be focused on more important and intricate tasks. With such vast possibility of applications possible, we provide one implementation of a head pose controlled system which is computationally light, fast and reliable as compared to other methods thereby allowing CPUs with a broader range of processing power to perform head pose estimation. We also demonstrate the system on a wireless robot car, controlled by head direction.

To achieve head pose estimation, we use a hybrid method by using geometric/algebraic methods with the Dlib face landmark tacker which detects the locations 68 landmark points on the given face. This ensure that the system is stable bad faces and noise which is common for geometric algorithms which only localize only face regions and boundary points. The geometric method involves using the POSiT (Pose from orthographic scaling with Iterations) algorithm on some of the 2D coordinates of the points supplied by the face landmark tracker.

We demonstrate the algorithm on a robot car run by a NodeMcu IC microcontroller and the L293D motor driver IC. It is connected wirelessly to the laptop via TCP/IP. It moves forwards, backwards, left and right depending on whether the user is looking up, down, left and right respectively. With further modification, it can be made to move according to the degree of head tilt also. Our system works well in most situations and works best with good lighting.

The robot responds to head direction commands in real time with only a 0.1s delay and has sufficient range.

This system works reliably and the robot executes motion for each direction continuously. This shows that a handsfree control system can be implemented with this approach. Using a reliable face tracker has increases stability. This system can be used to build surveillance drones, wheelchairs and safety systems

***Keywords: Python, Opencv, Dlib, POSiT, NodeMcu.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Contents** | | | | | |
|  | | | | | Page No |
| Acknowledgement | | | |  | i |
| Abstract | | | |  | ii |
| List Of Figures | | | |  | iii |
| List Of Tables | | | |  | vi |
|  | | | | | |
| **Chapter 1** | | | **INTRODUCTION** | | **1** |
|  | **1.1** | Introduction | | | 1 |
|  | **1.2** | Motivation | | | 2 |
|  | **1.3** | Organization of Report | | | 3 |
|  | | | | | |
| **Chapter 2** | | | BACKGROUND THEORY and/or LITERATURE REVIEW | | **4** |
|  | **2.1** | Lierature review | | | 4 |
|  | **2.2** | Summarized outcome of literature review | | | 6 |
|  | **2.3** | Theoretical discussions | | | 7 |
|  | **2.4** | Conclusion | | | 10 |
|  | | | | | |
| **Chapter 3** | | | **METHODOLOGY** | | **11** |
|  | **3.1** | Methodology | | | 11 |
|  | **3.2** | Tools used & conclusion | | | 28 |
|  | | | | | |
| **Chapter 4** | | | **RESULT ANALYSIS** | | **29** |
|  | **4.1** | Result analysis | | | 29 |
|  | **4.2** | Significance of result obtained | | | 36 |
|  | | | | | |
| **Chapter 5** | | | **CONCLUSION AND FUTURE SCOPE** | | **37** |
|  | **5.1** | Brief summary of work | | | 37 |
|  | **5.2** | Work Conclusion | | | 37 |
|  | **5.3** | Future Scope of Work | | | 38 |
|  | | | | | |
| **REFERENCES** | | | | | **39** |
| **PROJECT DETAILS** | | | | | **40** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Title** | **Page No** |
| 1.1 | Flow of project model | 3 |
| 2.1 | Visualization of how hybrid models combine different methods | 6 |
| 3.1 | Illustrating roll, yaw and pitch of head in terms of X, Y, Z | 11 |
| 3.2 | Flowchart highlighting various processes in the project | 13 |
| 3.3 | Flowchart showing how Dlib face landmark model is trained | 14 |
| 3.4 | Locations of the 68 landmark points detected by Dlib | 15 |
| 3.5 | Real world & camera coordinate systems | 16 |
| 3.6 | How iterations of POS form the POSIT algorithm | 21 |
| 3.7 | IC L293D pinout | 24 |
| 3.8 | Circuit diagram of robot | 25 |
| 3.9 | Circuit diagram of experimental robot built with RaspberryPi | 27 |
| 4.1 | Original sample image | 30 |
| 4.2 | Output of Dlib shape landmark predictor | 30 |
| 4.3 | Screenshot of final pose being identified by software | 31 |
| 4.4 | Image categories of custom dataset | 32 |
| 4.5 | Results of table 4.2 in graph form. Faces detected v/s faces missed | 33 |
| 4.6 | Results of table 4.2 in graph form. Faces detected v/s correct poses | 34 |
| 4.7 | NodeMcu controlled robot used | 35 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Title** | **Page No** |
| 4.1 | Specifications of custom dataset | 29 |
| 4.2 | Results of testing our algorithm on the MIT-postest dataset | 32,33 |

**CHAPTER 1**

**INTRODUCTION**

From an early age, people display the ability to quickly and effortlessly interpret the orientation and movement of a human head, thereby allowing one to infer the intentions of others who are nearby and to comprehend an important nonverbal form of communication. The ease with which one accomplishes this task belies the difficulty of a problem that has challenged computational systems for decades. In a computer vision context, head pose estimation is the process of inferring the orientation of a human head from digital imagery. It requires a series of processing steps to transform a pixel-based representation of a head into a high-level concept of direction. Like other facial vision processing steps, an ideal head pose estimator must demonstrate invariance to a variety of image-changing factors. These factors include physical phenomena like camera distortion, projective geometry, multisource non-Lambertian lighting, as well as biological appearance, facial expression, and the presence of accessories like glasses and hats.

* 1. *Brief present-day scenario:*

At the coarsest level, head pose estimation applies to algorithms that identify a head in one of a few discrete orientations, e.g. a frontal versus left/right profile view. At the fine (i.e. granular), a head pose estimate might be a continuous angular measurement across multiple degrees of freedom (DOF). A system that estimates only a single DOF, perhaps the left to right movement is still a

head pose estimator, as is the more complex approach that estimates a full 3D orientation and position of a head, while incorporating additional DOF including movement of the facial muscles and jaw [1]. In the context of computer vision, head pose estimation is most commonly interpreted as the ability to infer the orientation of a person’s head relative to the view of a camera. More rigorously, head pose estimation is the ability to infer the orientation of a head relative to a global coordinate system, but this subtle difference requires knowledge of the intrinsic camera parameters to undo the perceptual bias from perspective distortion.

In this project, we present a method for estimating 3D pose of head from a 2D RGB camera and we also how we control a robot to emulate the pose detected. To increase stability as compared to contemporary approaches we employ geometric/ algebraic methods along with face landmark tracker for stability and accuracy.

* 1. *Motivation*

Previous head pose estimation techniques like [2] use only geometric algebra to find head pose based on orientation of the 2D points in the image which is not reliable. Geometric methods when used alone, may be offset completely by noise and unexpected feature. More modern techniques like [3] which use template matching to achieve head pose estimation are accurate but they are time and processor intensive. AAM based models like [4] are fairly fast and reliable, but they need a very accurate database and high computing power, which makes it not feasible for portable systems.

In this project, we present a method for estimating head pose on a low powered system with a low-resolution camera. This system is fast, reliable, accurate and needs less computing power. We use geometric/algebraic methods to allow usage on lower powered systems, but to increase stability, we also use a face feature tracker to derive face points from image for the geometric methods to work on. Since the algorithm relies on an independent base face landmark tracker, it makes it reliable and accurate. We demonstrate the performance of the algorithm on a wireless controlled robot car.

* 1. *Objective:*

Objective of the work is to present a light, reliable and accurate method for head pose estimation and to emulate the output on a moving robot. Our system should be fast and be able to run on all systems. The robot will change direction according to the direction the user is looking up. If he looks up, the robot moves forward, down it moves backward and so on. It allows hands free control of a robot.

* 1. *Target specifications:*

The final system will calculate head orientation of the user in degrees in X, Y, Z coordinates in 3D space and will classify the head direction in terms of up, down, left and right based on the calculated 3D coordinates. Input is taken from laptop webcam, and the command signal is passed to the robot wirelessly which executes the corresponding motion according to the head direction in real time.

****­­­­­­**

**Wi-Fi Command signal sent over TCP/IP**

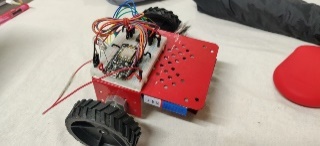
**­­****

Fig 1.1: Flow of project model

* 1. *Organization of report:*

In the first chapter we introduce the area of work and the broad flow of the report. The second chapter covers the related work in the area and literature survey. In the third chapter we explain the methodology and how the face landmark tracker, POSiT and the wireless robot come into play.

In the fourth chapter, we analyse the results obtained from testing and comment on the accuracy of the system. We provide concluding remarks and explore future scope in the last chapter.

**CHAPTER 2**

**BACKGROUND THEORY**

This chapter discusses the various methods that have been used to estimate head pose over time. We explain why our method is the best fit for our case, we also explain the advantages of our method selection over the other commonly available solutions. We also explain some of the basic terminologies and components involved in the project.

We propose another method for controlling a robot from head pose. Our system uses footage from a RGB camera. The image of the user’s head is captured and the head pose is identified and the command signal is sent wirelessly to the motor control hardware which executes the corresponding output accordingly. The robot moves forward if user is looking up, moves backward if looking down, moves left if looking left and moves right when looking right accordingly.

We choose to use hybrid and geometric/algebraic methods to achieve head pose estimation. We use the orthographic scaling technique to project the 2D points captured by the “Dlib face landmark detector” into the 3D plane and find the rotation and translation vectors with a rough reference 3D model. The values of the rotation and translation vectors of the head are used to derive the head pose in terms of X, Y, Z coordinates. To improve the accuracy and reliability of geometric methods, we use the “Dlib shape predictor” face landmark tracker. This ensures that we have reliable face information to calculate pose from so we don’t have to worry about the presence of high frequency noise causing non-convergence of angle values. Using only geometric algebraic methods make the system very sensitive to high frequency noise, so we employ hybrid methods by using a face landmark tracker

The head pose, on being tracked gets sent wirelessly to the hardware control. The hardware is driven by a NodeMcu IC, which is connected to the laptop wirelessly over a TCP/IP. Upon receiving the direction command from the laptop, the NodeMcu IC

drives hardware accordingly to execute the corresponding motion.

*2.1 Literature review*

Many head pose estimation techniques have been tried over the years, some of the first attempts Ire made in 1998 with appearance template models [8]. Since then the iodometry has come a long way. With current hardware, complex deep learning-based models can be trained with a variety of datasets like AFW,300w, HELEN. Amazon has also built integrated “Amazon Rekognition” software in its “AWS cloud tier” making the engine accessible to developers for testing their own databases for commercial applications.

Considering the feasibility of deployment, I chosen methods that will require less computational complexity, less hardware cost and easy adaptability from face to face. [4]

Eight categories that describe the conceptual approaches that have been used to estimate head pose:

* Appearance template methods like compare a new image of a head to a set of exemplars (each labelled with a discrete pose) in order to find the most similar view.
* Detector array methods train a series of head detectors each attuned to a specific pose and assign a discrete pose to the detector with the greatest support.
* Nonlinear regression methods use nonlinear regression tools to develop a functional mapping from the image or feature data to a head pose measurement.
* Manifold embedding methods seek low-dimensional manifolds that model the continuous variation in head pose. New images can be embedded into these manifolds and then used for embedded template matching or regression.
* Flexible models fit a nonrigid model to the facial structure of each individual in the image plane. Head pose is estimated from feature-level comparisons or from the instantiation of the model parameters.
* Geometric methods need the location of features such as the eyes, mouth, and nose tip to determine pose from their relative configuration.
* Tracking methods recover the global pose change of the head from the observed movement between video frames.
* Hybrid approaches combine one or more of the aforementioned methods to estimate pose, as illustrated by example in Fig. 10. These systems can be designed to overcome the limitations of any one specific head pose category. A common embodiment is to supplement a static head pose estimation approach with a tracking system. The static system is responsible for initialization and the tracking system is responsible for maintaining pose estimates over time. If the tracker begins to drift, the static system can reinitialize the track. This method yields the high accuracy of pure tracking approaches without initialization and drift limitations. Many successful combinations have been presented by mixing an automatic geometric method with point tracking [4]

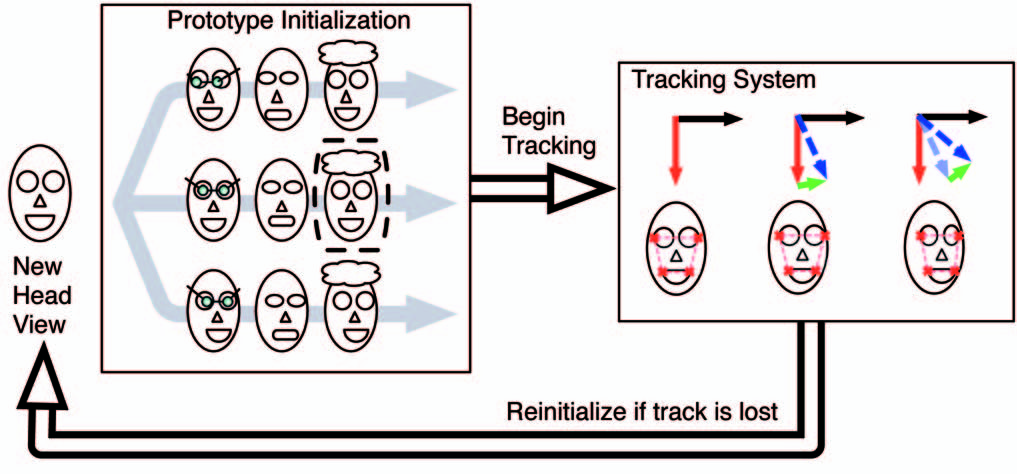


Fig: 2.1 Visualization of how hybrid methods combine different methods. Hybrid methods combine one or more approaches to estimate pose. This image is an example of an appearance template method combined with a point tracking system.

*2.2 Summarized outcome of literature review*

There are many pose estimation algorithms, but we use point-based pose estimation using projective geometry. While manifold embedding methods and flexible models can benefit pose estimation, they can be excessive in computation time. The methods with the manifold embedding and flexible methods require initialization, and reliance on such information may fail in some situations such as sudden head movements. Point-based algorithm allows fast estimation with no temporal information so it can also be applied to still images. Our point-based pose estimation algorithm uses the Pose from Orthography and Scaling with ITerations (POSIT) algorithm, and we extend on its limitations for head pose estimation purposes. POSIT with landmark is a hybrid pose estimation algorithm of algebraic/geometric algorithms and optimizing algorithms. Algebraic/geometric algorithms tend to fall under numerical instabilities in the presence of noises, which is frequent with facial feature extractions. Optimizing algorithms are robust against noises, but an initial guess is required that heavily affects the performance in convergence. POSIT accommodates advantages from both approaches and achieve good speed and accuracy. Robustness is increased by initial guess with a direct linear transformation, and error minimization is performed through iterations through optimization algorithms like Levenberg- Marquardt optimization.

The input is captured by the laptop webcam. We use the “Dlib shape predictor” pretrained model to detect face landmarks from the input image. Using this information, we use the POSiT algorithm which does orthographic scaling to project the 2D points in the 3D plane. The points derived from it are compared to the reference 3D points to calculate rotation and translation vectors which give the head pose coordinates in X, Y, Z.

This method relies only on the RGB camera of the laptop and thereby reduces computational complexity, allowing lower powered systems to run. Also, the computational time involved is far less than in template matching or AAM methods

The command signal is then sent to the hardware control wirelessly over a TCP/IP connection. The robot hardware is controlled by a NodeMcu IC which connects to internet via Wi-Fi.

The setup consists of a NodeMcu IC, an L293 motor driver, two DC motors and two 9volt batteries.

*2.3 Theoretical discussions*

*2.3.1 Opencv*

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

*2.3.2 Dlib shape predictor:*

The *Face Landmark Detection* algorithm offered by Dlib is an implementation of the **Ensemble of Regression Trees (ERT)**presented in 2014 by Kazemi and Sullivan [3]. The face detector is made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme.

This technique utilizes simple and fast feature (*pixel intensities differences*) to directly estimate the landmark positions. These estimated positions are subsequently refined with an iterative process done by a *cascade of regressors.*The regressors produces a new estimate from the previous one, trying to reduce the alignment error of the estimated points at each iteration. The algorithm is blazing fast, in fact it takes about 1–3ms (on desktop platform) to detect (align) a set of 68 landmarks on a given face.

It is trained on the iBUG-300w dataset, which is a combination of the datasets LFPW, AF, HELEN and XM2VTS.

*2.3.3 POSiT:*

It is possible to estimate the 3D rotation and translation of a 3D object from a single 2D photo, if an approximate 3D model of the object is known and the corresponding points in the 2D image are known. A common technique for solving this has recently been "POSIT", where the 3D pose is estimated directly from the 3D model points and the 2D image points, and corrects the errors iteratively until a good estimate is found from a single image. Most implementations of POSIT only work on non-coplanar points (in other words, it won't work with flat objects or planes).

*2.3.4 Levenberg-Marqardt optimization:*

The Levenberg-Marquardt curve-fitting method is actually a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic and finding the minimum of the quadratic. The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. This project illustrates the use of software to solve the nonlinear least squares curve-fitting problem that arises from POSiT using the Levenberg optimization.

*2.3.5 NodeMcu:*

The NodeMCU is an open source Lua based firmware for the ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif NON-OS SDK.The firmware was initially developed as is a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now be run on any ESP module.

NodeMCU Dev Kit has Arduino like Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc. ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

When ESP8266 hosts the application, and when it is the only application processor in the device,

it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and to minimize the memory requirements. Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any microcontroller-based design with simple connectivity through UART interface or the CPU AHB bridge interface.

NodeMcu’s on-board processing and storage capabilities allow it to be integrated with the sensors

and other application specific devices through its GPIOs with minimal development up-front and

minimal loading during runtime. With its high degree of on-chip integration, which includes the antenna switch balun, power management converters, it requires minimal external circuitry, and

the entire solution, including front-end module, is designed to occupy minimal PCB area. Sophisticated system-level features include fast sleep/wake context switching for energy efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS,

LCD interference mitigation.

*2.3.6 TCP/IP:*

The Transmission Control Protocol/Internet Protocol, is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet). The entire internet protocol suite- a set of rules and procedures- is commonly referred to as TCP/IP, though others are included in the suite

TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination. TCP/IP requires little central management, and it is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network.

The two main protocols in the internet protocol suite serve specific functions. TCP defines how applications can create channels of communication across a network. It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.

IP defines how to address and route each packet to make sure it reaches the right destination. Each gateway computer on the network checks this IP address to determine where to forward the message.

*2.4 Conclusion:*

Compared to traditional methods, our method is more reliable, faster and requires less computing power. AAM and template matching methods are almost as accurate but they require more power and computation time. Geometric/algebraic methods are computationally light, but they get put off by presence of high frequency noise in images, which is very common in low resolution cameras. So to increase the stability, we use a pretrained face landmark tracking model-“Dlib shape predictor” this means that the algorithm does not get offset with noisy pictures, spectacles etc. since the tracker traces the points, and the pose estimation algorithm only needs to calculate from the location of the points detected by the tracker. -

**CHAPTER 3**

**METHODOLOGY**

This chapter discusses the components and concepts involved and the working of the project. We also talk about how the various systems used are connected and their role.

*3.1 Methodology*

We use Fig. 3.2 to illustrate the flow of the various systems involved in the project. It starts with the laptop webcam capturing the image of the user. The Dlib shape predictor is a popular face landmark detection tool used, we use the pretrained model available in the Dlib package for python. This return a vector containing the 2D locations (x, y coordinates) of 68 landmark points on the face.

The POSiT algorithm is implemented by the cv2.SolvePnP function. It uses some of the 68 points detected along with reference 3D points of face features as suggested by [3]. It iterates until convergence and returns the rotation and translation matrix of the object (i.e. face) which gives us the Euler angles i.e. X, Y, Z coordinates.

* X value determines pitch; i.e. degree of. up & down motion
* Y value determines yaw; i.e. degree of left & right motion
* Z value determines roll; i.e. degree of vertical tilt

The values in X, Y, Z degrees tell the head direction to a fine degree. We classify direction as up, down, left & right.

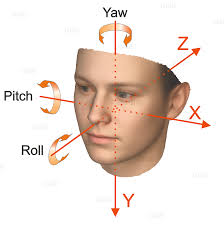


Fig 3.1 Illustrating roll, yaw and pitch of head in terms of X, Y, Z

Fig 3.2: Flowchart highlighting the various processes in the project

*3.1.1 Face landmark detection:*

The *Face Landmark Detection* algorithm offered by Dlib is an implementation of the **Ensemble of Regression Trees (ERT)**presented in 2014 by Kazemi and Sullivan [3]. The face detector is made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme.

This technique utilizes simple and fast feature (*pixel intensities differences*) to directly estimate the landmark positions. These estimated positions are subsequently refined with an iterative

Face bounding box detected by HoG based detector

**Step 1**

Another cascade refines the points derived by the first

Cascade of Regressors learns points based on gradient boosting

**Step 2**

Convergence reached?

**NO**

**Step 3**

**YES**

Face landmark points array generated

Fig 3.3: Flowchart showing how the Dlib face landmark model is trained.

process done by a *cascade of regressors.*The regressors produces a new estimate from the previous one, trying to reduce the alignment error of the estimated points at each iteration. The algorithm is blazing fast, in fact it takes about 1–3ms (on desktop platform) to detect (align) a set of 68 landmarks on a given face. The landmark points are as shown in Fig 3.3

It is trained on the iBUG-300w dataset, which is a combination of the datasets LFPW, AF, HELEN and XM2VTS. We choose to track 14 points out of the 68 points detected by the tracker as shown in Fig 3.1. These are 17, 21, 22, 26, 36, 39, 42, 45, 31, 35, 48, 54, 57 and 08 in the figure below. We apply the pose estimation algorithm to these 14 points to project them into the 3D plane and find the rotation and translation vectors of the points.

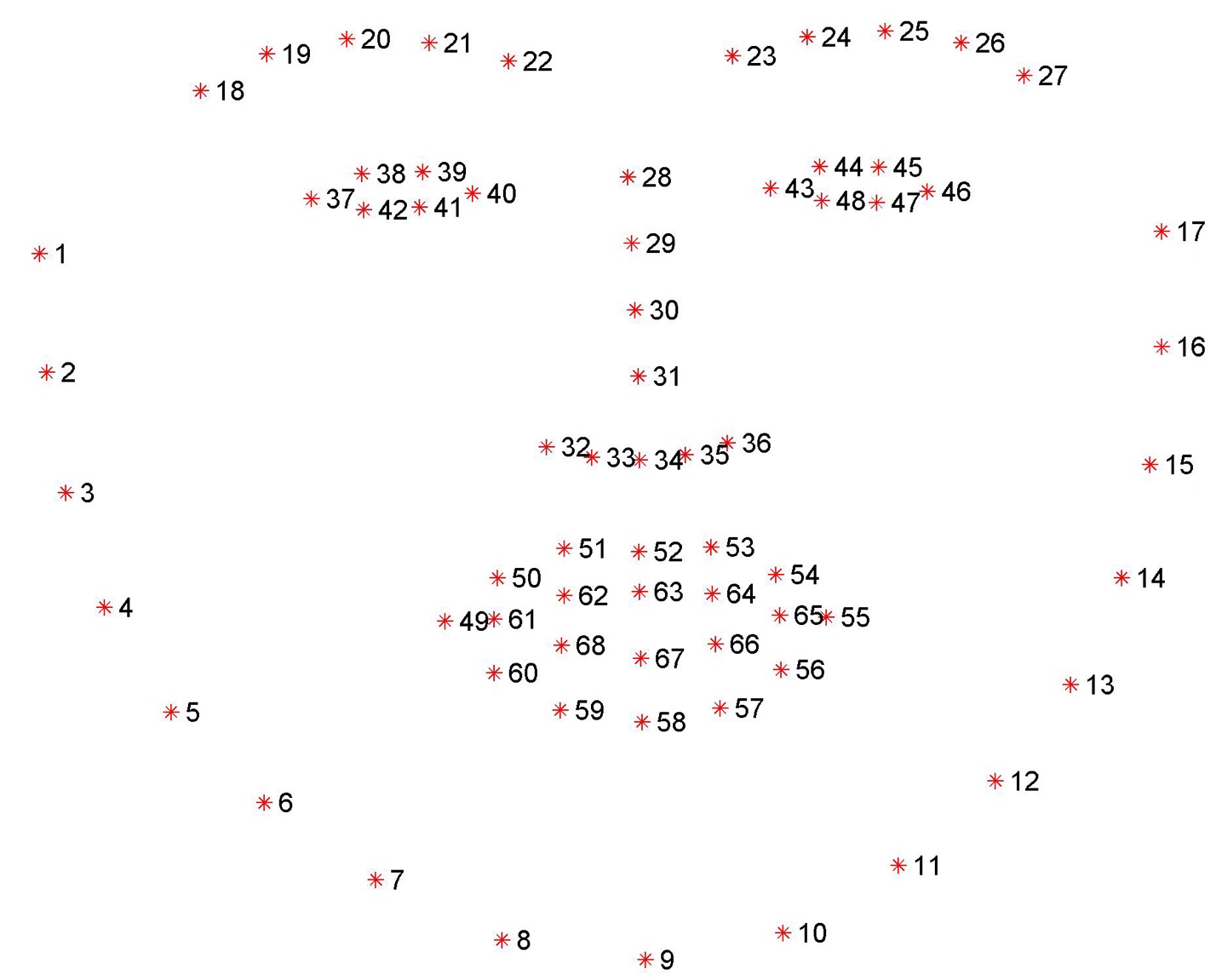


Fig 3.4: The locations of the 68 landmark points detected by the face landmark tracker- “Dlib shape predictor”

*3.1.2 General working of Pose estimation algorithms:*

There are several algorithms for pose estimation. The first known algorithm dates back to 1841. There are three coordinate systems in play. The 3D coordinates of the various facial features shown below in Fig 3.4 are in world coordinates. If we knew the rotation and translation (i.e. pose), we could transform the 3D points in world coordinates to 3D points in camera coordinates. The 3D points in camera coordinates can be projected onto the image plane (i.e. image coordinate system) using the intrinsic parameters of the camera (focal length, optical centre etc.).

Now we explain the image formation equation to understand how the coordinate systems in Fig 3.4 work. O is the centre of the camera and plane shown in the figure is the image plane. We are interested in finding out what equations govern the projection ρ of the 3D point P onto the image plane.

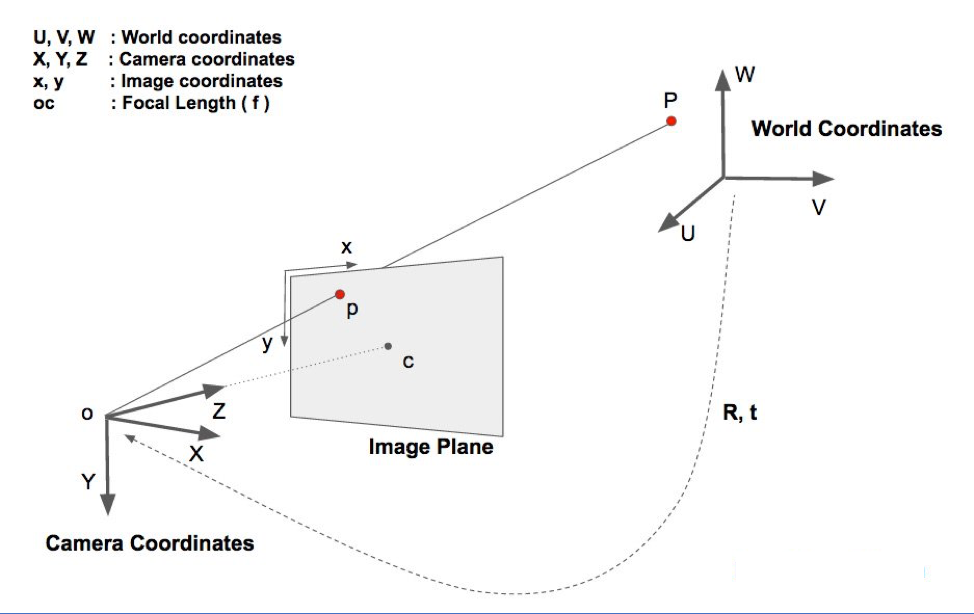
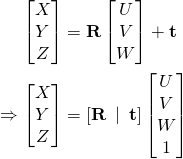
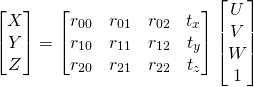
[](https://www.learnopencv.com/wp-content/uploads/2016/09/ImageFormationEquation.jpg)

Fig 3.5: The real-world coordinate system and camera coordinate system and how they are projected.

Let us assume we know the location U, V, W of a 3D point P in World Coordinates. If we know the rotation R (a 3×3 matrix) and translation “t” (a 3×1 vector), of the world coordinates with respect to the camera coordinates, we can calculate the location X, Y, Z of the point P in the camera coordinate system using the following equation (Eq 3.1).

 -Eq 3.1

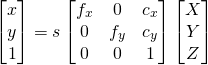
In expanded form, the above equation looks like this (Eq 3.2)

 -Eq 3.2

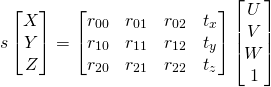
If we know sufficient number of point correspondences (i.e. X, Y, Z and U, V, W), the above is a linear system of equations where the rij and tx, ty, tz are unknowns and we can trivially solve for the unknowns. We know X, Y, Z only up to an unknown scale, and so we do not have a simple linear system.

*3.1.3 Direct Linear Transform:*

We do know many points on the 3D model (i.e. U, V, W), but we do not know (X, Y, Z). We only know the location of the 2D points (i.e. (x, y)). In the absence of radial distortion, the coordinates (x, y) of point P in the image coordinates is given by (Eq 3.3)

  -Eq 3.3

where, fx and fy are the focal lengths in the x and y directions, and (cx, cy) is the optical centre. In the equation s is an unknown scale factor. It exists in the equation due to the fact that in any image we do not know the depth. If you join any point P in 3D to the centre o of the camera, the point P, where the ray intersects the image plane is the image of P. Note that all the points along the ray joining the centre of the camera and point P produce the same image. In other words, using the above equation, we can only obtain (X, Y, Z) up to a scale s. As show in (Eq 3.4)

 -Eq 3.4

Fortunately, the equation of the above form can be solved using a method called Direct Linear Transform (DLT). DLT is used when there is a problem where the equation is almost linear but is off by an unknown scale.

*3.1.4 Levenberg-Marquardt Optimization:*

The DLT solution mentioned above is not very accurate because of the following reasons. First, rotation R has three degrees of freedom but the matrix representation used in the DLT solution has 9 numbers. There is nothing in the DLT solution that forces the estimated 3×3 matrix to be a rotation matrix. More importantly, the DLT solution does not minimize the correct objective function. Ideally, we want to minimize the reprojection error that is described below.

As shown in the equations 2 and 3, if we knew the right pose (R and t), we could predict the 2D locations of the 3D facial points on the image by projecting the 3D points onto the 2D image. In other words, if we knew R and t, we could find the point P in the image for every 3D point P.

We also know the 2D facial feature points (using Dlib or manual clicks). We can look at the distance between projected 3D points and 2D facial features. When the estimated pose is perfect, the 3D points projected onto the image plane will line up almost perfectly with the 2D facial features. When the pose estimate is incorrect, we can calculate a re-projection errormeasure the sum of squared distances between the projected 3D points and 2D facial feature points.

As mentioned earlier, an approximate estimate of the pose (R and t) can be found using the DLT solution. A naive way to improve the DLT solution would be to randomly change the pose (R and t) slightly and check if the reprojection error decreases. If it does, we can accept the new estimate of the pose. We can keep perturbing R and t again and again to find better estimates. While this procedure will work, it will be very slow. There are principled ways to iteratively change the values of R and t so that the reprojection error decreases. One of these methods is called the Levenberg-Marquardt optimization.

The primary application of the Levenberg–Marquardt algorithm (Eq 3.5) is in the least-squares curve fitting problem: given a set of {\displaystyle m}empirical datum pairs (xi,yi) {\displaystyle \left(x\_{i},y\_{i}\right)} of independent and dependent variables, find the parameters β{\displaystyle {\boldsymbol {\beta }}} of the model curve {\displaystyle f\left(x,{\boldsymbol {\beta }}\right)}f (xi,yi) {\displaystyle \left(x\_{i},y\_{i}\right)} so that the sum of the squares of the deviations {\displaystyle S\left({\boldsymbol {\beta }}\right)}S(β{\displaystyle {\boldsymbol {\beta }}}) is minimized:

 -Eq 3.5{\displaystyle {\hat {\boldsymbol {\beta }}}\in \operatorname {argmin} \limits \_{\boldsymbol {\beta }}S\left({\boldsymbol {\beta }}\right)\equiv \operatorname {argmin} \limits \_{\boldsymbol {\beta }}\sum *{i=1}^{m}\left[y{i}-f\left(x*{i},{\boldsymbol {\beta }}\right)\right]^{2},}

We describe the method for finding the pose of faces from a single image implemented. We

assume that we can detect and match in the image four or more coplanar feature points

(i.e. the eyes, the nose, and the mouth) of the faces, and that we know their relative geometry

on the faces.

The method combines two algorithms, the first algorithm: POS (Pose from Orthography and Scaling) approximates the perspective projection with a scaled orthographic projection and finds the rotation matrix and the translation vector of the faces by solving a linear system.

The second algorithm: POSIT (POS with ITerations) uses in its iteration loop the approximate pose found by POS in order to compute better scaled orthographic projections of the feature points, and then applies POS to these projections instead of the original image projections. POSIT converges to accurate pose measurements in a few iterations, POSIT can be used with many feature points at once for added insensitivity to measurement errors and image noise. Compared to classic approaches making use of Newtons method, POSIT does not require starting from an initial guess, and computes the pose using an order of magnitude fewer floating point operations, it may therefore be a useful alternative for real-time operation.

*3.1.5 POS algorithm:*

The method relies on linear algebra techniques and is iterative, but it does not require an initial pose estimate and does not require matrix inversions in its iteration loop. At the first iteration step, the method finds an approximate pose by multiplying an object matrix (which depends only on the distribution of feature points on the object and is precomputed) by two vectors (which depend only on the coordinates of the images of these feature points). The two resulting vectors, once normalized, constitute the first two rows of the rotation matrix, and the norms of these vectors are equal to the scaling factor of the projection, which provides the translation vector [1]. These operations amount to assuming that the involved image points have been obtained by a scaled orthographic projection (SOP in the following).

*3.1.6 POSIT algorithm:*

The next iterations apply exactly the same calculations, but with corrected image points. The basic idea is that since the POS algorithm requires an SOP image instead of a perspective image to produce an accurate pose, we have to compute SOP image points, using the pose found at the previous iteration. The process consists in shifting the feature points of the object. in the pose just found, to the lines of sight (where they would belong if the pose was correct) and obtains a scaled. orthographic projection of these shifted points.

This is the iterative algorithm POSIT. Four or five iterations are typically required to converge to an accurate pose. For N matching between object points and image points, POSIT requires around 24N arithmetic operations and two square root calculations per iteration. For 8 feature points and

four iteration steps, around 800 arithmetic operations and 8 square roots are needed. We use the OpenCV function solvePnP to estimate the 3d points of the 2d landmarks to. Give us the rotation and translation vectors and the Euler angles of the head. Y axis determines yaw i.e. left right motion, X determines pitch i.e. up down motion and Z determines roll i.e. tilt along vertical axis. For a specific range of values of X, Y, Z the program classifies pose as left, right, up and down. All of this is computed on the laptop.

Fig 3.5 shows the flow of the POSiT algorithm working simultaneously for 2 sets of a landmark feature point. Image point or landmark feature point refers to the chosen points from the 2D coordinates (x, y) of the face landmark points captured by the Dlib face landmark tracker. Coplanar image points are the 3D locations of the points chosen for reference. Focal length and image Centre array is set as default as suggested by the official OpenCV docs [6].

Zi refers to the image points, E (Epsilon) is the error from one iteration of the POS algorithm, the error between the actual image points and the projected points from the computed pose. This error E is minimized in the next iteration using Levenberg-Marquardt optimization. T, R refers to the final translation and rotation vectors of the object (i.e. Face). R1, R2 refers to the intermediate rotation matrix values from one iteration of POS. the process is shown running

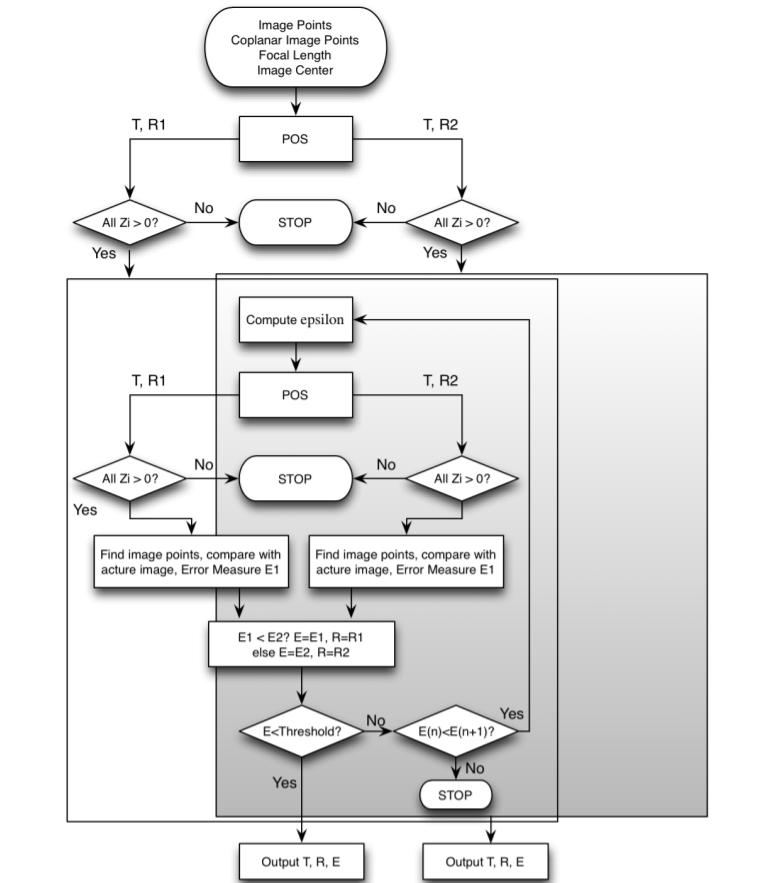


Fig 3.6 How iterations of POS form the POSiT algorithm

*3.1.7 TCP/IP:*

The Transmission Control Protocol/Internet Protocol, is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet). The entire internet protocol suite -- a set of rules and procedures -- is commonly referred to as TCP/IP, though others are included in the suite*.*

We assign the laptop as the server and the NodeMcu as the client. The NodeMcu contains code for directing wheel motion and setting up wireless connections, and the laptop send only the message containing the direction detected as the command.

*3.1.8 Hardware control:*

We use TCP/IP protocol to communicate between the NodeMcu and the laptop. The NodeMcu controls the motors and drives them according to the command received from the laptop. The setup consists of a NodeMcu, an L293D motor driver, two DC motors, two 9volt batteries. The specifications are attached below.

We choose NodeMcu Microcontroller for its widespread availability, and cost. It is a NodeMcu is an open source IoT platform.It includes Lua based firmware which runs on the ESP8266 Wi-Fi SoC. It costs 2$ and has 16 GPIO pins with PWM. available on all pins.

The NodeMcu is connected to the laptop via a TCP/IP connection. Both devices are connected to the internet via Wi-Fi. The NodeMcu continuously listens for commands from the laptop server. The laptop transmits the head pose direction currently being detected in real time, and the NodeMcu receives this signal and drives the motors according to the command sent in real time through its GPIO pins. The robot is as shown in Fig 3.3

NodeMcu features:

 802.11 b/g/n protocol

 Wi-Fi Direct (P2P), soft-AP

 Integrated TCP/IP protocol stack

 Integrated TR switch, balun, LNA, power amplifier and matching network

 Integrated PLL, regulators, and power management units

 +19.5dBm output power in 802.11b mode

 Integrated temperature sensor

 Supports antenna diversity

 Power down leakage current of < 10uA

 Integrated low power 32-bit CPU could be used as application processor

 SDIO 2.0, SPI, UART

 STBC, 1×1 MIMO, 2×1 MIMO

 A-MPDU & A-MSDU aggregation & 0.4s guard interval

 Wake up and transmit packets in < 2ms

 Standby power consumption of < 1.0mW (DTIM3)

*3.1.9 L293D IC:*

293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive in either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that it can control two DC motor. In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below in Fig 3.6 is the pin diagram of a L293D motor controller.

mVCC is the voltage that it needs for its own internal operation 5v; L293D will not use this voltage for driving the motor. For driving the motors, it has a separate provision to provide motor supply VSS (V supply).  L293d will use this to drive the motor. It means if you want to operate a motor at 9V then you need to provide a Supply of 9V across VSS Motor supply.The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel. Since it can drive motors Up to 36v hence you can drive pretty big motors with this l293d.

VCC pin 16 is the voltage for its own internal Operation. The maximum voltage ranges from 5v and upto 36v.

The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

Each IC is connected to the GPIO pins of the NodeMcu. The 5v IC power supply, enable, input 1&2 and ground are connected to NodeMcu GPIO. The 9volt power supply for the motor is supplied by a battery.

The motors used are of 100rpm max speed and rated at 12volts. The permissible input is between 5-12v, and current is 200mA to 1.5A. we use a castor wheel for support, thereby two motors suffice for moving in all directions.

The circuit diagram of the robot hardware is as shown in Fig3.7

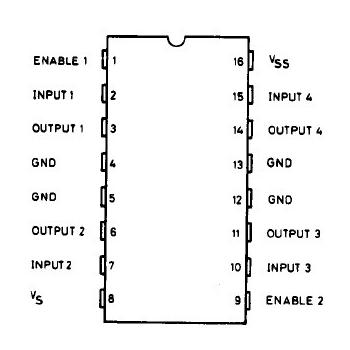


Fig 3.7 IC L293D pinout diagram

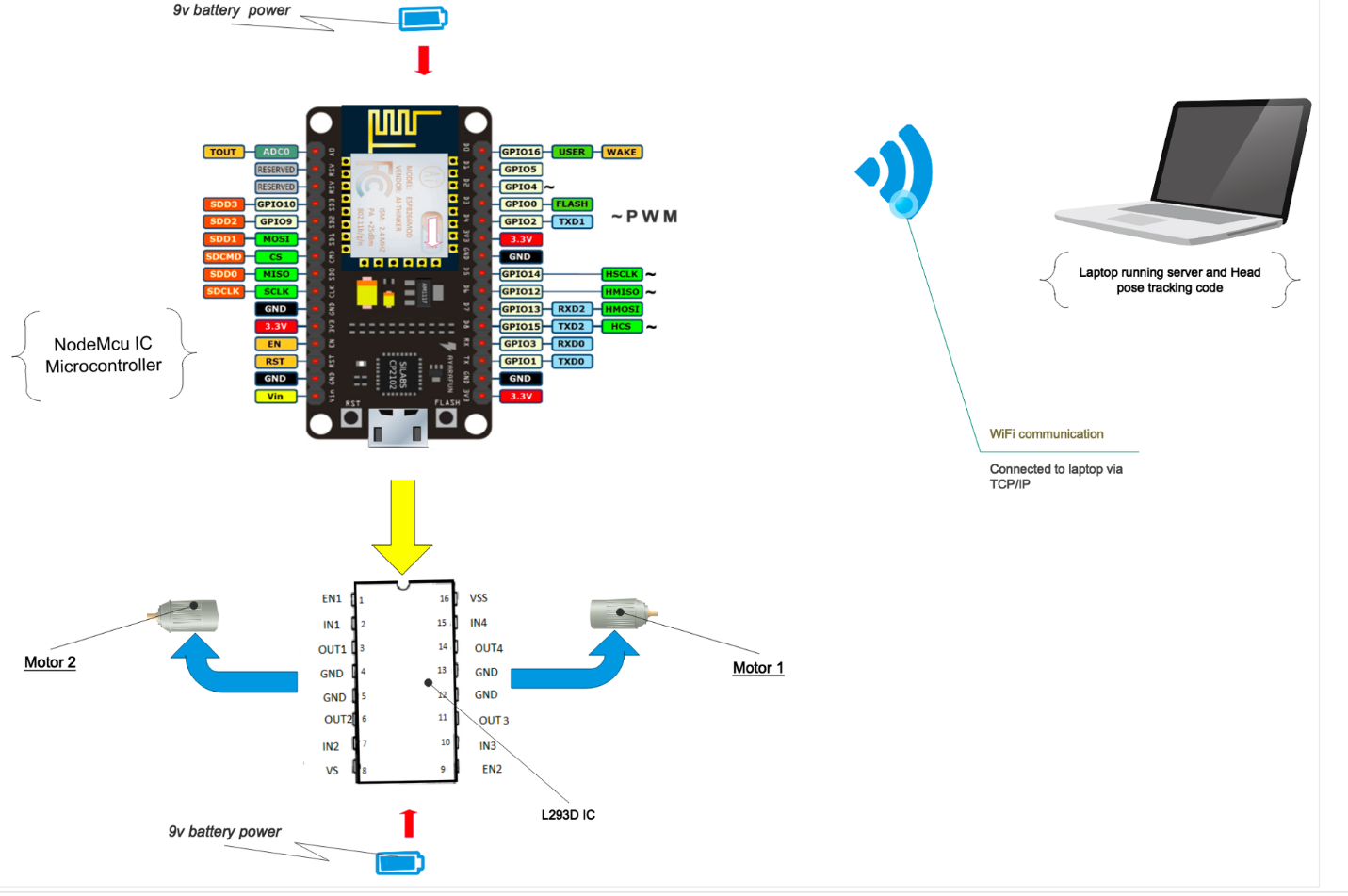


Fig 3.8: Circuit diagram of robot. Connected to laptop via WiFi

*3.1.10 Experimental robot: RaspberryPi:*

To fine tune choice of. Hardware for applications that involve accuracy, we also tested a system built on a different platform, the RaspberryPi. The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then.The latest model has a quad-core 1.4GHz CPU with 1GB RAM. All over the world, people use Raspberry Pis to learn programming skills, build hardware projects, do home automation, and even use them in industrial applications. The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).

We decided to experiment with it for the following reasons

* 40 pin GPIO pins available in Raspberrypi as compared to 16 in NodeMcu allowing the option to plug in much more devices.
* RaspberryPi has an ethernet port, HDMI port, 3.5mm audio jack, USB 3.0 ports and USB 2.0 ports. Whereas NodeMcu has only a Microusb port. This means we can connect peripherals to the RasberrpyPi, allowing much more ways for devices to interact with it.
* RaspberryPi has far more RAM and processing. Power, it runs its own Linux based OS. So possibilities are limitless

Reasons for not using RaspberryPi:

* Expensive hardware compared to NodeMcu. Almost 30 times the cost
* Requires more power to drive, making the robot bulky unnecessarily.

The circuit diagram of the robot made with the RaspberryPi is as shown in Fig 3.8. It uses two L293D motor driver ICs to split the load and increase heat dissipation. It also uses a power bank to power the RaspberryPi.

For this hardware, we tested the range, max speed, delay, accuracy of motion:

Max speed- 7.2Kmph

Delay in response- 0.11s

Range of network connection- 23 meters

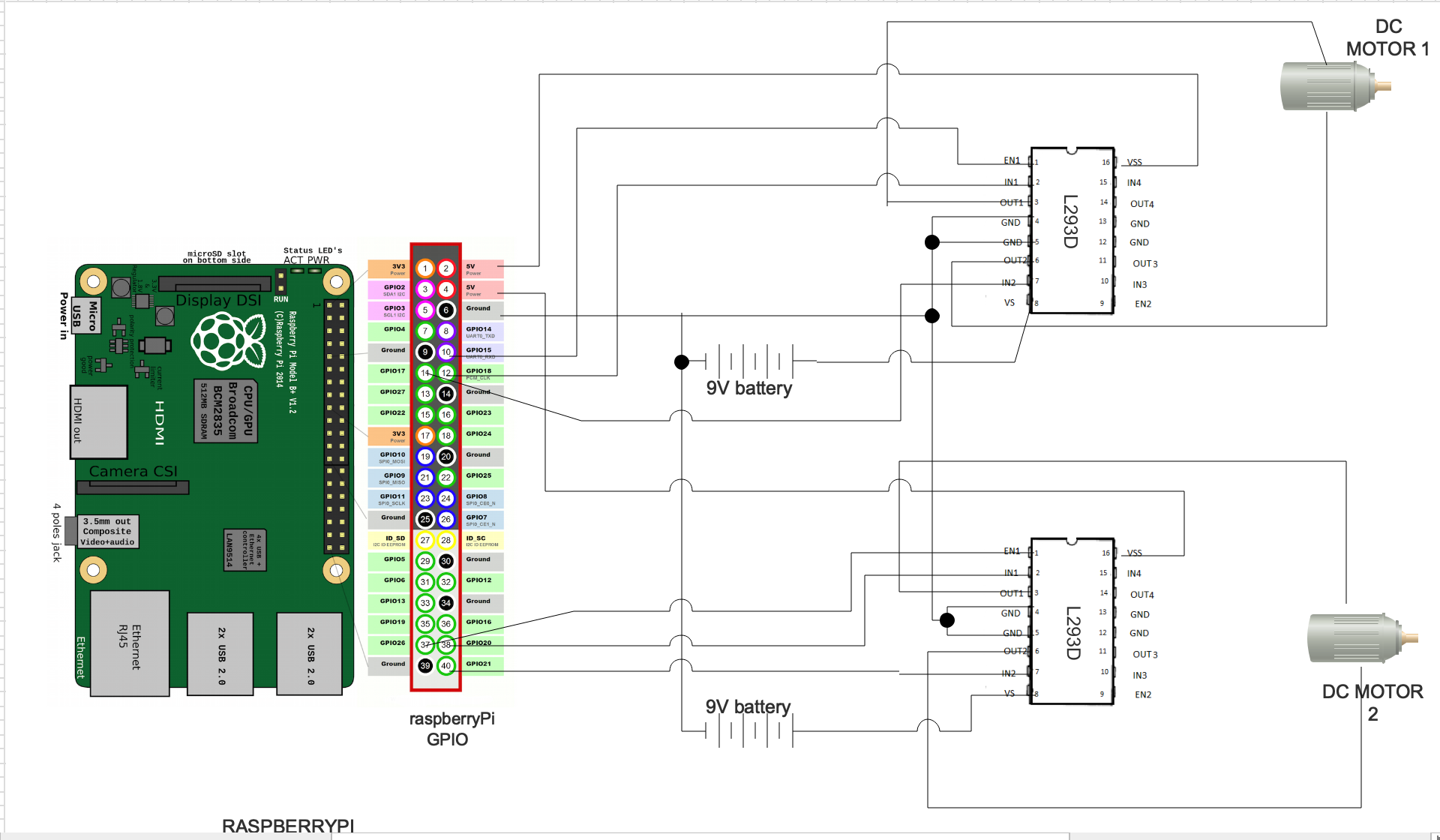
Robot range with fresh battery- 1875 meters (approx.)

Battery capacity- 450mAh x 2

Capacity of raspberry pi power supply- 6hrs

The robot has similar operational performance as compared to the one built using the NodeMcu, but it is bulkier and has more power requirements. The added functionality of using the RaspberryPi does not come in handy in our use case scenario, so we use NodeMcu instead for its efficiency.

­­

Fig 3.9: Circuit diagram of robot built with RaspberryPi showing GPIO and IC connections

*3.2 Tools used & Conclusion:*

The code is implemented in python version 3.7 with the OpenCV 3.0 and Dlib 19.1.0 packages installed.

The system used to run and test the software has the specs:

* CPU: Intel i5 8th gen
* RAM: 8gb
* Webcam resolution: 1920x1080p
* GPU: Nvidia mx250
* OS: Windows 10 x64

On taking the user image, we apply the “Dlib shape predictor” function provided in the Dlib 19.1.0 package. This returns a matrix containing locations of 68 tracked face feature points. Upon transforming the matrix, we apply the POSiT algorithm.

To implement POSiT, we use the function cv2.SolvePnP which implements DLT (direct linear transform followed by Levenberg-Marquardt optimization as mentioned in sections 3.1.3 & 3.1.4 respectively. The function is supplied with the landmark points from the Dlib face landmark tracker. Also, we supply a reference 3D points of face features as suggested by [6]. The camera calibration parameters are set to default as prescribed by the OpenCV official documentation.

The code for a TCP/IP server is also running on the laptop, the NodeMcu robot runs as a TCP client and connects to the laptop over Wi-Fi and listens for commands. When the laptop recognizes a head direction from left, right, up & down, the command is sent to the robot which moves accordingly in real time.

**CHAPTER 4**

**RESULT ANALYSIS**

In this chapter we document the results obtained from the testing of our hardware and software. We tabulate the accuracy of our software and the conditions used for testing.

*4.1 Result analysis*

The output of the various steps of the algorithm is described here. The original image is as shown in Fig 4.1. The white noise is removed and so the pixel intensities are adjusted to differentiate face features and background. the output of the Dlib face landmark tracker is shown in Fig 4.2. It shows 68 points tracked on the face; we only use some of those.

The final output of the algorithm, with the Euler angles is as shown in Fig 4.3 It also shows the face landmark points tracked, the face bounding box bordering the face region.

We tested. The software and the hardware under controlled conditions. For testing the head pose recognition, we first supply it with 100 images from the “MIT-posetest” dataset. This contains a small mix of pictures people’s faces in 10 categories as listed in the Table 4.2. examples of image of each category is shown in Fig 4.4. It consists of 100 JPEG images, with a resolution of 1280x720 pixels in the RGB space. The images also attached with the correct head pose values as a benchmark to compare our results. We loop the algorithm to run on all 100 faces Consecutively, we find the mean error, category by category for all the images.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No. of images** | **No. of categories** | **Image format used** | **Min. image size** | **Max. image size** | **Colour space** |
| 100 | 10 | .jpg | 310 kB | 1214kB | RGB |

Table 4.1: Specifications of our custom dataset



Fig 4.1: Original sample image

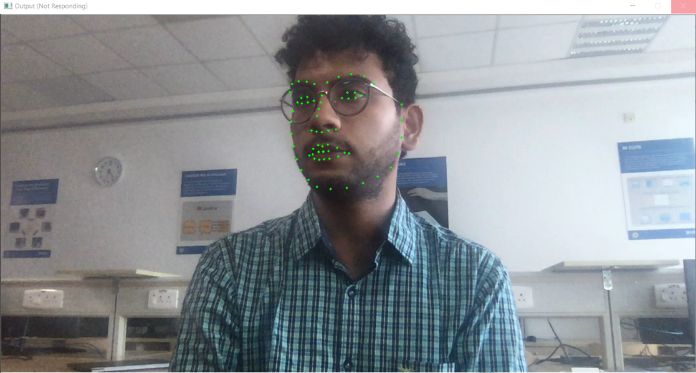
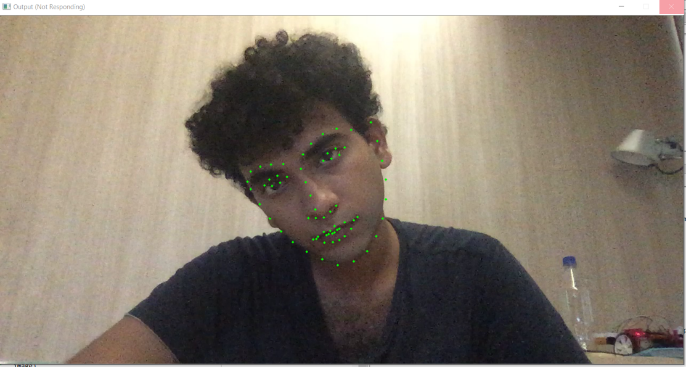


Fig 4.2 Output of Dlib shape landmarks predictor. Green points represent the face landmarks detected

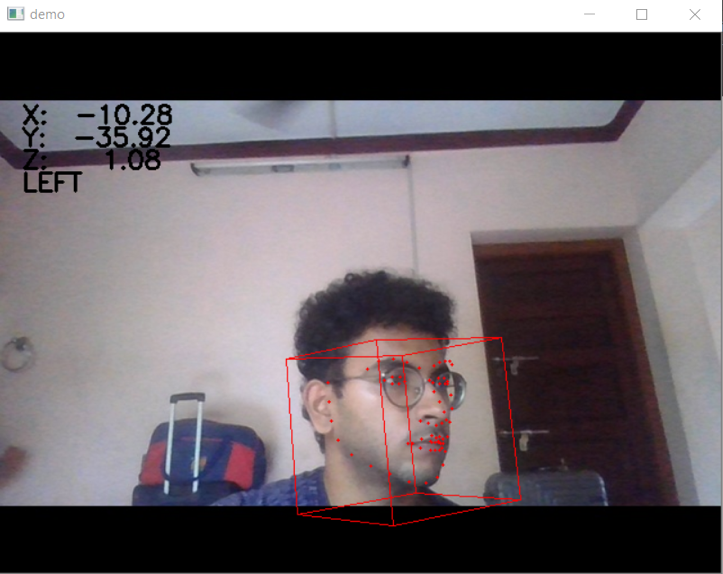
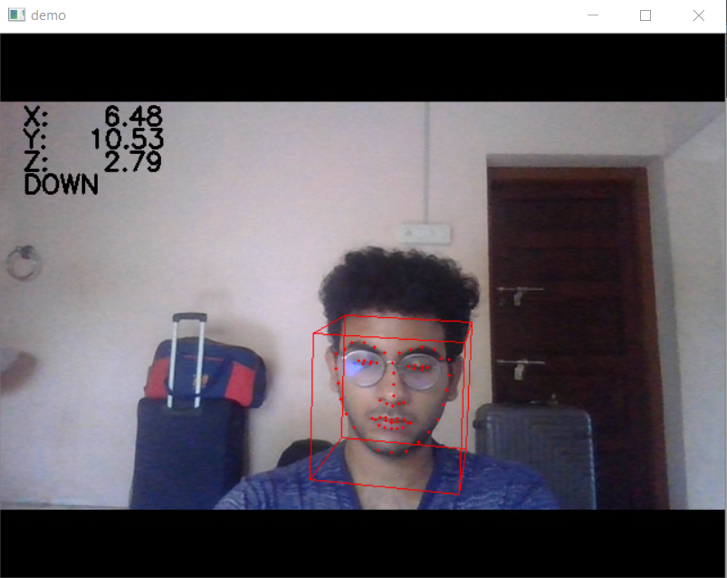
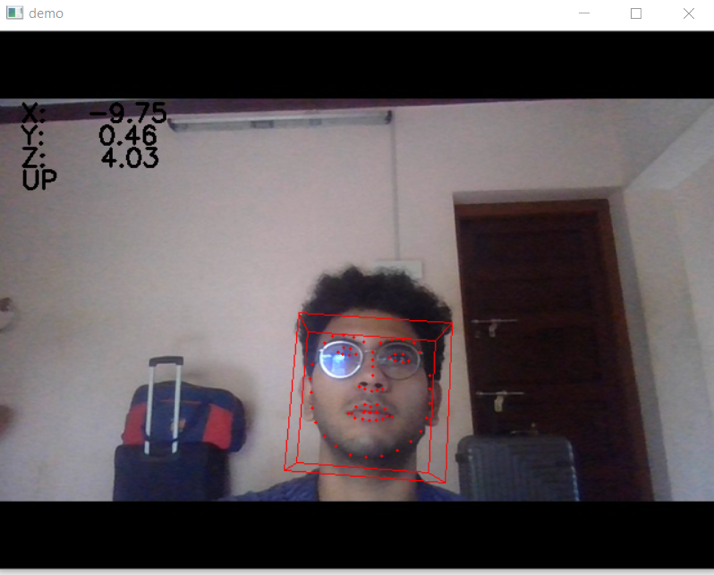


Fig 4.3: Screenshot of head pose in image being identified by software. Clockwise: down, right, left, up.

Mean yaw error is the error about X axis. i.e. left-right motion. Mean pitch error is the error about Y axis i.e. up-down motion. Mean roll error is the error about Z axis i.e. head tilt motion. Our code is optimized to run to detect not more than 70 degrees head tilt in any direction.

No. of faces detected is the number of faces detected by the face landmark tracker (Dlib Shape predictor) from the 10 images of the particular category.

No. of correct detections is the number of poses detected and classified accurately by direction (left, right, up and down).



Fig 4.4: Image categories of custom dataset, the categories are mentioned in table 4.2 respectively

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Serial**  **No.** | **Type of image** | **Mean yaw error** | **Mean roll error** | **Mean pitch error** | **No. of faces detected** | **No. of faces missed** | **No. of correct detections** | **No. of incorrect detections** |
| 1 | Good indoor lighting | ±15.8 | ±22.3 | ±32.6 | 8 | 2 | 6 | 2 |
| 2 | Outdoor environment | ±14.3 | ±20.2 | ±29.3 | 9 | 1 | 7 | 2 |
| 3 | With eyewear | ±19.4 | ±27.5 | ±36.2 | 7 | 3 | 4 | 3 |
| 4 | Indoor with glare | ±23.6 | ±29.1 | ±39.4 | 5 | 5 | 4 | 1 |
| 5 | With face hair | ±17.7 | ±23.8 | ±34.6 | 7 | 3 | 5 | 2 |
| 6 | Fair skin tones | ±18.0 | ±23.2 | ±35.5 | 7 | 3 | 5 | 2 |
| 7 | Dark skin tonnes | ±16.9 | ±23.1 | ±33.0 | 8 | 2 | 6 | 2 |
| 8 | With face deformities | ±20.4 | ±25.2 | ±36.3 | 6 | 4 | 4 | 2 |
| 9 | Part of face covered by hair, cloth etc. | ±22.7 | ±27.2 | ±37.2 | 5 | 5 | 4 | 1 |
| 10 | Detecting one face in foreground among multiple | ±22.5 | ±34.5 | ±41.2 | 5 | 5 | 4 | 1 |

Table 4.2: Results of testing the algorithm on custom dataset derived from MIT posetest dataset

Fig 4.5: Results of Table 4.2 in graph form. It shows the no. of faces detected v/s the number of faces missed (i.e. error rate). Category numbers are the same as in Table 4.2 respectively.

Fig 4.6 Results of Table 4.2 in graph form. Plotted values are No. of Faces detected and no of correct pose identified out of those. Category numbers same as in Table 4.2 respectively.

We draw the following analysis from Fig 4.5, Fig 4.6 & Table 4.2:

* We get the best results in good outdoor lighting where the greatest number of faces were able to be detected and the greatest number of correct poses were recognized. As it should be.
* Skin colour and face hair don’t cause the system to deviate from the ideal results by much. This is expected as the Dlib shape predictor is trained on the “iBUG 300w” dataset which accounts for all these variations. It looks for the boundary of the face, eyes, nose and mouth. As long as these parts are visually distinguishable and not obscured.
* The cases where the specimen wearing spectacles or sunglasses were slightly less accurate. The algorithm is sensitive to the parts of the face being blocked or obscured. As the face landmark detector relies on detecting 10 points around the eyes. Eyewear changes the shape of the eyes as it appears to the camera, thus the boundary points of the eye cannot be detected.
* The accuracy of detection for the cases where considerable parts of the face were obscured by clothing or deformities. This includes persons wearing a muslin, scarf, bandana, face mask etc. where the more than a quarter of the face is covered. Cases of face deformities include, eye missing, face warped, mouth misaligned etc. since in these cases, the landmark points on face are misaligned as compared to the generic 3D head model we used. So, when the landmark points are wrong, their 3D projections calculated from POSiT will be wrong, so the pose is wrong.
* For cases where a person is in crowded groups, the face landmark detector struggles to identify the face in the foreground since we have not provided depth data. On detecting the face and landmarks, it managed to find pose with decent accuracy, at least in cases where there was not too much glaring light.

We also tested the reliability of the hardware (Fig 4.6):

* Range: approx.: 3000m
* NodeMcu battery life: 20hrs
* Max speed: 8kmph
* Motor battery life (for average use): 6hrs

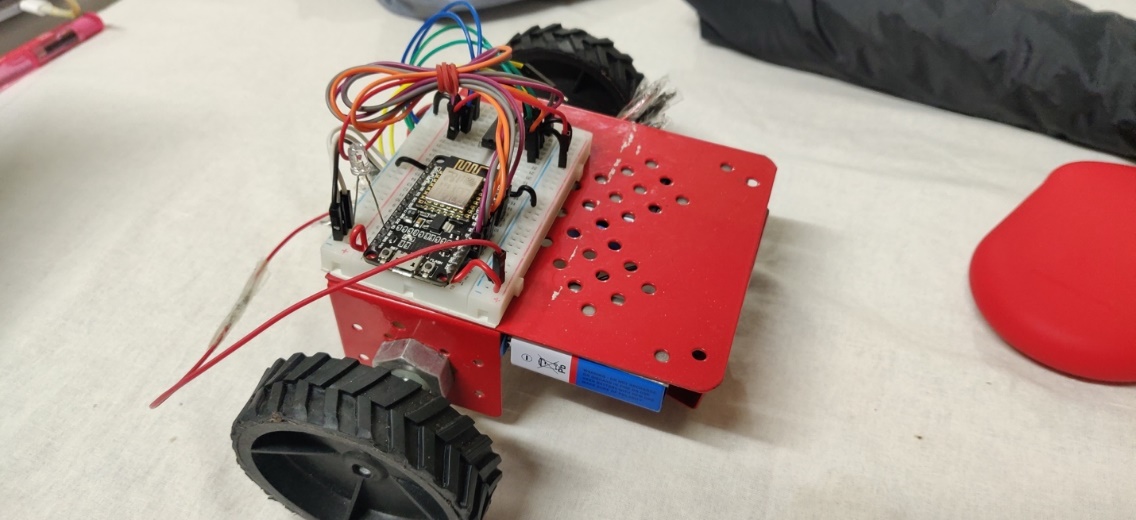
**

Fig 4.7: NodeMcu controlled robot used

*4.2 Significance of result obtained:*

The system has reasonable accuracy in most settings and has good accuracy in good lighting. the system is reliable and recognised left, right, up and down. However, when the situation gets tricky, it falters. Improper or uneven lighting cause the face landmark tracker to miss multiple points thereby returning an empty matrix, so no pose can be calculated. Covering parts of the face also causes the. same issue. When there are multiple faces at same distance, the algorithm does not understand which one to track, so as a safety precaution it doesn’t return any value.

The hardware works fine with no heating issues. The batteries Require charging about every 8hrs. When within range, the NodeMcu has a response time of about 0.1s. this can be. Improved by connecting the NodeMcu via HTTP, instead of sending packets using TCP/IP to connect over the internet.

The system works with reasonable accuracy provided lighting conditions are proper and provided face abnormalities are not too significant. The user need not remove eyewear, but needs to remove any cloth covering the face. The system is person independent, so it allows any user to use it without the need of any setup. We loop the algorithm over the frames of a video to achieve real time head pose estimation.

**CHAPTER 5**

**CONCLUSION AND FUTURE SCOPE OF WORK**

*5.1 Brief summary of work*

Our goal was to build a system which uses head pose tracking to control hardware movements. The aim was to recognize user’s head direction in X, Y, Z coordinates accurately and to send the commands to a robot wirelessly. We had to achieve this with low hardware cost and computational power, while still maintaining stability and speed.

We use a hybrid method for head pose tracking []. We build on geometric/algebraic methods, which are fast and computationally but are easily offset but imperfections in user faces and also by high frequency noise. So, we use a separate face tracker to detect the face points for the geometric algorithm to calculate pose from these points. These points are called face landmark points or face landmarks. This means that the pose estimation algorithm is provided with accurate data coordinates.

We use the popular face landmark pertained model “Dlib shape predictor” as the face landmark tracker. To calculate pose, we use the POSiT (Pose from orthographic scaling and iterations) algorithm which uses DLT (direct linear transform) followed by Levenberg-Marquardt optimization. This gives us the Euler angles of head direction in X, Y, Z coordinates.

For implementing the algorithm, we build hardware controlled by the head pose direction. Using the NodeMcu IC microcontroller, we build a two wheeled robot. The microcontroller is connected to the laptop wirelessly over a TCP/IP connection. Depending on the head direction sent by the laptop, the NodeMcu drives the motors accordingly in real time. We use the L293D motor driver IC to drive the motors.

*5.2 Work Conclusions:*

Our system detects head pose in real time fairly accurately in most cases. It works particularly well with good lighting and clear faces. The connection to the robot is fast and reliable too and the robot functions as expected.

The user image is captured by the laptop. Our algorithm calculates head angles in 3D coordinates. The NodeMcu microcontroller driving the robot is connected to the laptop wirelessly and receives head direction command signals from the it continuously. If the user tis looking up, the robot moves forward, if he is looking down it moves backwards, if looking left if turns left, and turns right when looking right accordingly.

This system can be used to wirelessly control hardware like robotic arms, wheelchairs and cars. The big advantage is that this allows hands-free control, which allows disabled people to operate the same hardware without difficulty.

*5.3 Future scope of work:*

Gesture technology is new and upcoming in recent years with. several applications already being implemented in largescale. It allows a much more natural way to control with hardware and leaves the hands of the user free to control other functions thereby increasing the overall functionality of the existing hardware tremendously.

*5.3.1 Head pose controlled wheelchair:*

A wheelchair controlled only by head direction can be used by quadriplegic patients, patients with Parkinson’s disease, amputees, paralysis etc. head pose based control is the most natural way for the patients to control movements, since the direction of their gaze tends to be the same as the direction they are moving in.

The computationally light hardware allows for cheap large-scale implementation for patients, hospitals, old age homes etc.

*5.3.2 Head* *pose controlled Drone*:

In places hard to reach, where humans can’t go we have to send a surveillance or reconnaissance drone. These drones are controlled remotely, and their movements and functionality are limited and not smooth. Integrating head pose based control, frees the user’s hands to control other movements. Thereby increasing the speed of communication with the robot dramatically.

**REFERENCES**

[1] Bertok, Kornel & Sajó, Levente & Fazekas, Attila. (2011). A robust head pose estimation method based on POSIT algorithm. Argumentum. 7. 348-356.

[2] Won Kim, Woo & Hwang, Jinkyu & Lee, Sangyoun. (2011). Automatic head pose estimation from a single camera using projective geometry. Proc. IEEE Int. Conf. Information, Communications and Signal Processing. 10.1109/ICICS.2011.6173539.

[3] One Millisecond Face Alignment with an Ensemble of Regression Trees  
*Vahid Kazemi, Josephine Sullivan*; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1867-1874

[4] Head Pose Estimation in Computer Vision: A Survey

Erik Murphy-Chutorian, Student Member, IEEE, and Mohan Manubhai Trivedi, Fellow, IEEE; IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 4, APRIL 2009

[5] Model-Based Object Pose in 25 Lines of Code DANIEL E DEMENTHON AND LARRY S. DAVIS; International Journal of Computer Vision, 15, 123-141 (1995)

[6] Martins, Pedro & Batista, Jorge. (2008). Monocular Head Pose Estimation. 5112. 357-368. 10.1007/978-3-540-69812-8\_35.

[7]www.uk.mathworks.com

[8] B. Ma, W. Zhang, S. Shan, X. Chen, and W. Gao, “Robust Head Pose Estimation Using LGBP,” Proc. 18th Int’l Conf. Pattern Recognition, pp. 512-515, 2006.

[9] I. Matthews and S. Baker, “Active Appearance Models Revisited,” Int’l J. Computer Vision, vol. 60, no. 2, pp. 135-164, 2004.

[10] T. Moeslund, A. Hilton, and V. Kru ̈ ger, “A Survey of Advances in Vision-Based Human Motion Capture and Analysis,” Computer Vision and Image Understanding, vol. 104, no. 2, pp. 90-126, 2006

PROJECT DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| *Student Details* | | | |
| **Student Name** | **SAI TARUN MADHIRA** | | |
| Register Number | 150907046 | Section / Roll No | B/07 |
| Email Address | Saitarun.madhira@gmail.com | Phone No (M) | 8019644194 |
|  | | | |
| *Project Details* | | | |
| **Project Title** | **Head-Pose Controlled robot** | | |
| Project Duration | 6 months | Date of reporting | 20/09/2019 |
|  |  | | |
| *Organization Details* | | | |
| **Organization Name** | **Manipal Institute of Technology** | | |
| Full postal address with pin code | Udupi - Karkala Rd, Eshwar Nagar, Manipal, Karnataka 576104 | | |
| Website address | www.manipal.edu | | |
|  |  | | |
|  |  | | |
| *Internal Guide Details* | | | |
| **Faculty Name** | **Dr. Ramya S. Lokesh** | | |
| Full contact address with pin code | Dept of E & C Engg, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA | | |
| Email address | Ramya.lokesh@manipal.edu | | |



