2D/3D POSE ESTIMATION USING POSENET

A project report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Engineering

In

COMPUTER ENGINEERING

Submitted by

Pradnya Krishnanath Borkar (Roll No. 7682)

Marilyn Mathew Pulinthitta (Roll No. 7662)

Mrunal Vijay Kapure (Roll No. 7643)

Under the guidance of

Mrs. Ashwini Pansare
(Assistant Professor, Computer Engineering Department)

Department of Computer Engineering

Fr.Conceicao Rodrigues College of Engineering

Bandra, Mumbai-400 050

Year: 2018-2019

This work is dedicated to my family. I am very thankful for their motivation and support.

Internal Approval Sheet

CERTIFICATE

This is to certify that the project entitled **"2D/3D Pose Estimation Using Posenet"** is a bonafide work of **Pradnya Krishnanath Borkar (7682), Marilyn Mathew Pulinthitta (7662), Mrunal Vijay Kapure (7643)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of "**Bachelor of Engineering"** in "**Computer Engineering" .**

Mrs.Ashwini Pansare

Supervisor/Guide

Dr. Sunil Surve	**Dr. Srija Unnikrishnan**
Head of Department	Principal

Approval Sheet

Project Report Approval

This project report entitled '**2D/3D Pose Estimation Using Posenet'** by **Pradnya Krishnanath Borkar, Marilyn Mathew Pulinthitta, Mrunal Vijay Kapure** is approved for the degree of **Bachelor of Engineering.**

Examiners

1.________________________

2.________________________

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Pradnya Krishnanath Borkar (Roll No. 7682) ________________

Marilyn Mathew Pulinthitta (Roll No. 7662) ________________

Mrunal Vijay Kapure (Roll No. 7643) ________________

Date:

Abstract

Leaning dance/ yoga takes a lot of dedication. User has to put in a lot of money and time in it. The objective of the Match-pose system is to provide user a platform to learn dance and yoga poses without putting in a lot of money. The system should be available to the user anytime he/she wants to unlike the training institutes which put time constraints on the learner. Also the Match-pose aims to provide learning facilities for more than one user on the same screen so that multiple users can learn a pose simultaneously.

Pose estimation refers to computer vision techniques that detect human figures in images and video, so that one could determine, for example, where someone's elbow shows up in an image. In computer vision and robotics, a typical task is to identify specific objects in an image and to determine each object's position and orientation relative to some coordinate system. This information can then be used, for example, to allow a robot to manipulate an object or to avoid moving into the object. The combination of position and orientation is referred to as the pose of an object, even though this concept is sometimes used only to describe the orientation. Exterior orientation and translation are also used as synonyms of pose. The image data from which the pose of an object is determined can be either a single image, a stereo image pair, or an image sequence where, typically, the camera is moving with a known velocity. The objects which are considered can be rather general, including a living being or body parts, e.g., a head or hands.

PoseNet is a machine learning model that allows for Real-time Human Pose Estimation. PoseNet can be used to estimate either a single pose or multiple poses. The system Match-pose makes use of PoseNet in order to locate the key points in human body and forms skeleton to match the pose.

Acknowledgments

We have great pleasure in presenting the report on **"2D/3D Pose Estimation Using Posenet"**. We take this opportunity to express our sincere thanks towards the guide **Mrs.Ashwini Pansare**, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with her during our visits to department.

We thank Prof. Sunil Surve, Head of Computer Dept., Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

Pradnya Krishnanath Borkar (Roll No. 7682)
Marilyn Mathew Pulinthitta (Roll No. 7662)
Mrunal Vijay Kapure (Roll No. 7643)

Date:

Contents

Abstract **iv**

List of Figures **viii**

1 Introduction **1**

1.1 Pose Estimation **1**

1.2 Pose Estimation Associated with Human Body. **2**

1.2.1 17 Keypoints Estimation **2**

1.2.2 Posenet for 17 keypoints Estimation **3**

1.3 Objectives. **5**

1.4 Motivation **6**

2 Literature Review **7**

2.1 Deep Learning for Pose Estimation **7**

2.2 Pose Estimation as DNN Based Regressions **8**

3 Problem Statement **11**

3.1 Drawbacks of Current Approach **11**

3.2 Solution to Above Problem **11**

4 Project Description **12**

4.1 Overview of Project **12**

4.2 Architechture Diagram **13**

4.3 UML Diagrams **15**

4.3.1 DFD Diagrams **15**

4.3.2 Use Case Diagram **16**

5 Technologies used and Requirements **18**

5.1 Technologies **18**

5.1.1 HTML **18**

5.1.2 CSS **18**

5.1.3 JavaScript **19**

5.1.4 PHP **19**

5.1.5 XAMPP Server. **19**

5.2 Hardware and Software Requirements **20**

5.2.1 Hardware requirement **20**

5.2.2 Software Requirement **20**

6 Implementation Details and Output **21**

6.1 Implementation code. **21**

6.1.1 Home Page (index.html) **21**

6.1.2 Image Gallery (gallery.php) **22**

6.1.3 Displaying Webcam o/p and user prompt (camera.html). **23**

6.1.4 JS associated with camera.html (camera.js) **26**

6.1.5 Estimating Image Keypoints and skeleton plotting (ImgEst.js). . . . **32**

6.1.6 Calculation of angles and comparison of poses(PoseCompare.js). . **34**

6.1.7 Changing colour of skeleton on pose match **40**

6.2 SnapShots **44**

7 Future Enhancements and Conclusion **48**

7.1 Future Enhancements **48**

7.2 Conclusion **48**

8 References **49**

List of Figures

Fig No.	Name	Pg.no
1.2.1.1	17 Keypoints	2
1.2.2.1	Confidence Score of keypoints	5
1.4.1	Motivation: Move Mirror	6
2.1	Identifying articulated joints	7
2.2	Cascade of Pose Regressors	9
2.3	DeepPose vs other approaches	10
4.2.1	Architecture Diagram	13
4.3.1.1	Usecase Diagram	15
4.3.2.1	DataFlow Diagram	16
6.2.1	HomePage	43
6.2.2	About Page	44
6.2.5	Instruction Page	45
6.2.6	Image Gallery	45
6.2.7	Webbrowser camera access permission	46
6.2.8	User Prompt and preview image	46

Chapter 1

Introduction

1.1 Pose Estimation

Pose estimation refers to computer vision techniques that detect human figures in images and video, so that one could determine, for example, where someone’s elbow shows up in an image. To be clear, this technology is not recognizing who is in an image—there is no personal identifiable information associated to pose detection. The algorithm is simply estimating where key body joints are. Ok, and why is this exciting to begin with? Pose estimation has many uses, from interactive installations that react to the body to augmented reality, animation, fitness uses, and more. In computer vision and robotics, a typical task is to identify specific objects in an image and to determine each object's position and orientation relative to some coordinate system. This information can then be used, for example, to allow a robot to manipulate an object or to avoid moving into the object. The combination of position and orientation is referred to as the pose of an object, even though this concept is sometimes used only to describe the orientation. Exterior orientation and translation are also used as synonyms of pose.The image data from which the pose of an object is determined can be either a single image, a stereo image pair, or an image sequence where, typically, the camera is moving with a known velocity. The objects which are considered can be rather general, including a living being or body parts, e.g., a head or hands. The methods which are used for determining the pose of an object, however, are usually specific for a class of objects and cannot generally be expected to work well for other types of objects.The specific task of determining the pose of an object in an image (or stereo images, image sequence) is referred to as pose estimation. The pose estimation problem can be solved in different ways depending on the image sensor configuration, and choice of methodology.

1.2 Pose Estimation Associated with Human Body

Pose Estimation of a human body refers to finding the position of the joints of the human body. Each of these joints are called keypoints. There are different number of keypoints considered in different methods to estimate the pose. We have implemented the 17 keypoints method for developing our system.

1.2.1 17 Keypoints Estimation

Here, 17 body joints are considered for pose estimation. These seventeen keypoints are seen the the diagram and table below:

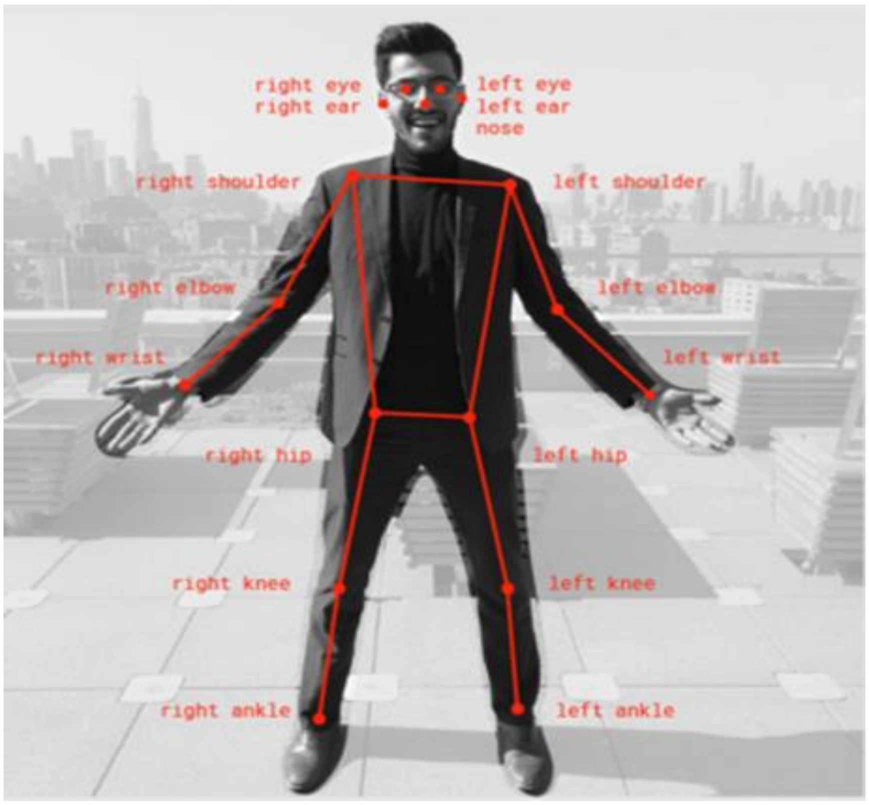


Fig. 1.2.1.1 : 17 keypoints

id	**part**
0	nose
1	left eye
2	right eye
3	left ear
4	right ear
5	left shoulder
6	right shoulder
7	left elbow
8	right elbow
9	left wrist
10	right wrist
11	left hip
12	right hip
13	left knee
14	right knee
15	left ankle
16	right ankle

Fig. 1.2.1.2 Keypoints table

1.2.2 Posenet for 17 keypoints generation

While many alternate pose detection systems have been open-sourced, all require specialized hardware and/or cameras, as well as quite a bit of system setup. With PoseNet running on TensorFlow.js anyone with a decent webcam-equipped desktop or phone can experience this technology right from within a web browser. And since it is an open sourced model, Javascript developers can tinker and use this technology with just a few lines of code. What's more, this can actually help preserve user privacy. Since PoseNet on TensorFlow.js runs in the browser, no pose data ever leaves a user's computer. 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.

At a high level pose estimation happens in two phases:

1. An input RGB image is fed through a convolutional neural network.
2. Either a single-pose or multi-pose decoding algorithm is used to decode poses, pose confidence scores, keypoint positions, and keypoint confidence scores from the model outputs.

But wait what do all these keywords mean? Let’s review the most important ones:

Pose—at the highest level, PoseNet will return a pose object that contains a list of keypoints and an instance-level confidence score for each detected person.

Pose confidence score—this determines the overall confidence in the estimation of a pose. It ranges between 0.0 and 1.0. It can be used to hide poses that are not deemed strong enough.

Keypoint—a part of a person’s pose that is estimated, such as the nose, right ear, left knee, right foot, etc. It contains both a position and a keypoint confidence score. PoseNet currently detects 17 keypoints illustrated in the above diagram 1.2.1.1.

Keypoint Confidence Score—this determines the confidence that an estimated keypoint position is accurate. It ranges between 0.0 and 1.0. It can be used to hide keypoints that are not deemed strong enough.

Keypoint Position—2D x and y coordinates in the original input image where a keypoint has been detected.

In this way our project makes use of posenet just for obtaining the 17 key points of the body as per the input image.

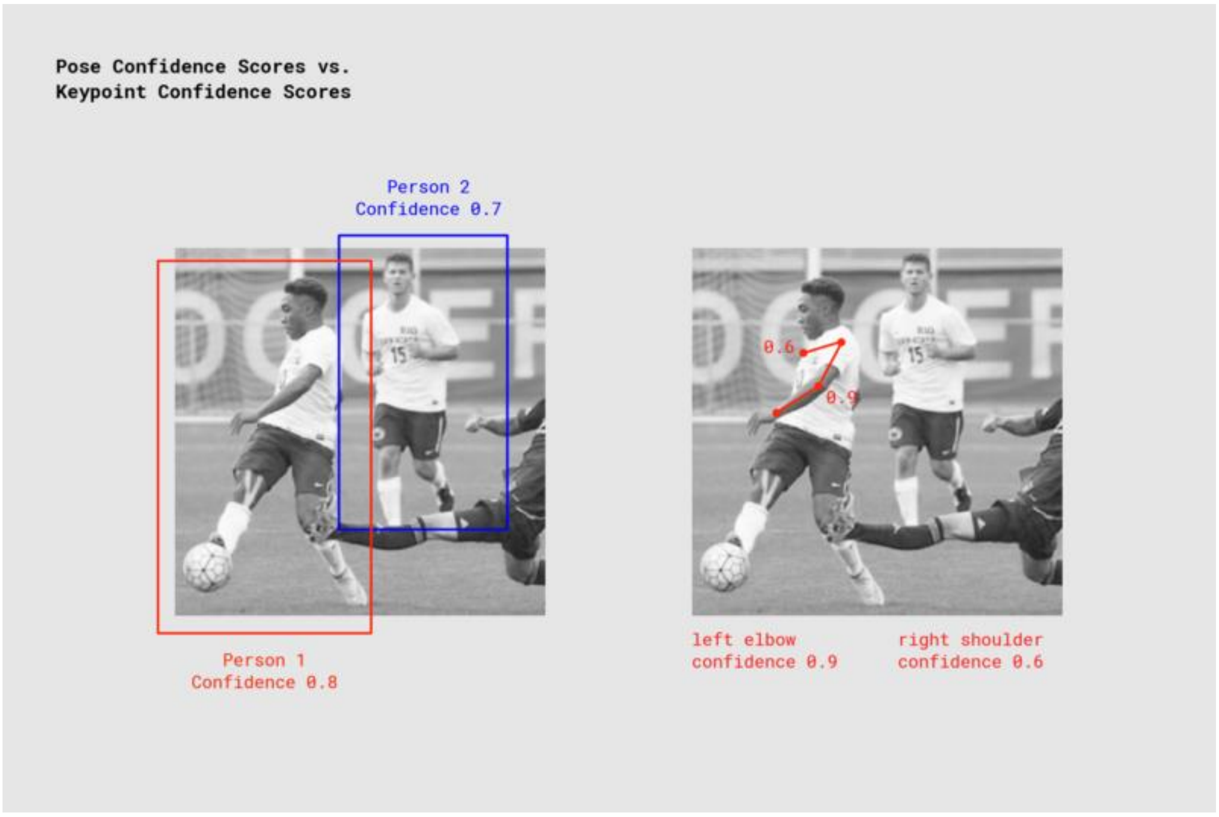


Fig. 1.2.2.1: Confidence score of Keypoints

1.3 Objectives

Our system is named MatchPose and it's name itself explains the objective or aim of our project. We aim to develop a system which will allow the user to replicate any pose from our image set without any third person assistance.Our system not just alerts the user on matching with the desired pose but also prompts the user of his deviations and improvements that the user can make to reach the desired pose. Learning to pose a certain way is very critical when it comes to art form like yoga or dance. This system will track the user pose in real time and will make the task of replicating poses easy and accurate. This system intends to overcome the deviations in poses that a human eye will not be able to detect. This system aims to get away from the traditional way of learning and is a new step towards technology driven learning.The aim of this system is to enable a user to match a desired pose with minimum hardware requirements and a webcam which has resolution greater than or equal to 2MP. This system aims to make the art of learning new poses innovative and fun, at the same time also aims providing the outcome with minimum requirements and prior setup.

1.4 Motivation

While we were researching through the new cutting edge technologies and new experiments, we came across the MoveMirror AI experiment initiated by google using Posenet. Their experiment uses a data set of thousands of images. The user can upload any image and all the images in the dataset that have similar poses will be displayed. We also decided to work in lines of this experiment but do something which is not just fun but also useful. That's when the idea of MatchPose came into our head. Wherein we take in users real time webcam feed and in real time itself match it with the desired pose.

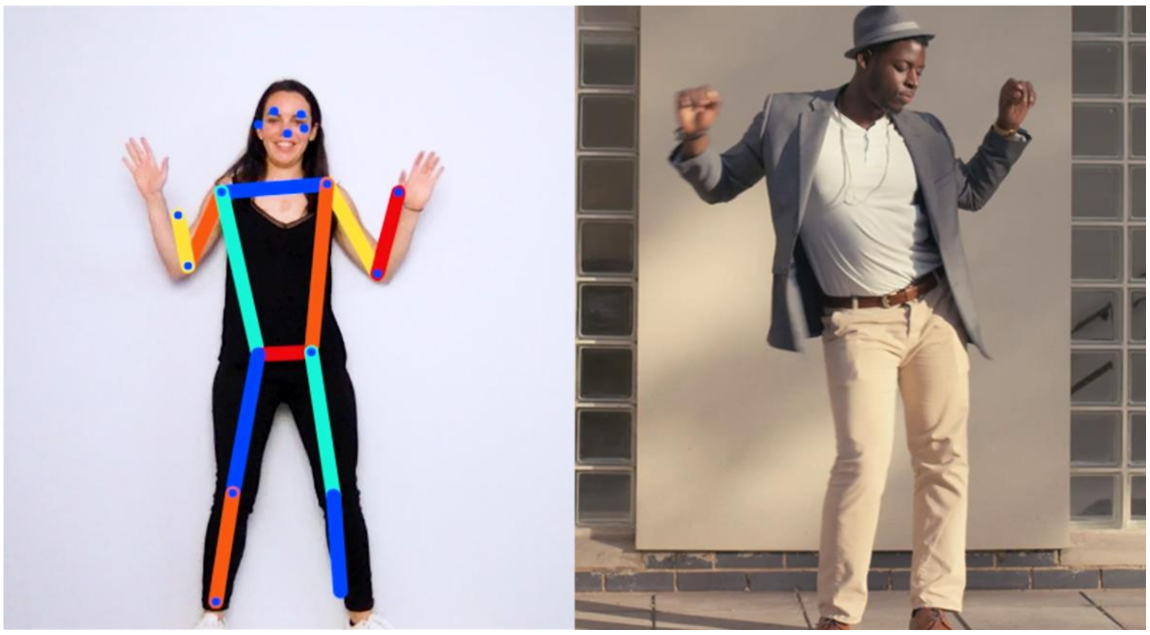


Fig. 1.4.1:Motivation: Move Mirror Experiment

Chapter 2

Literature Review

The problem of human pose estimation, defined as the problem of localization of human joints, has enjoyed substantial attention in the computer vision community. The challenge is to capture strong articulations, small and barely visible joints, occlusions.The main stream of work in this paper has been motivated mainly by the this challenge, the need to search in the large space of all possible articulated poses. For this purpose use of part-based models which naturally lend to model articulations has been proposed in this paper. There have been many other approaches for pose estimation, however all of them are achieved at the cost of limited expressiveness and use of local detectors, which reason in many cases about a single part, and most importantly by modeling only a small subset of all interactions between body parts. These limitations, are exemplified in this paper and methods reasoning about pose in a holistic manner have been proposed.

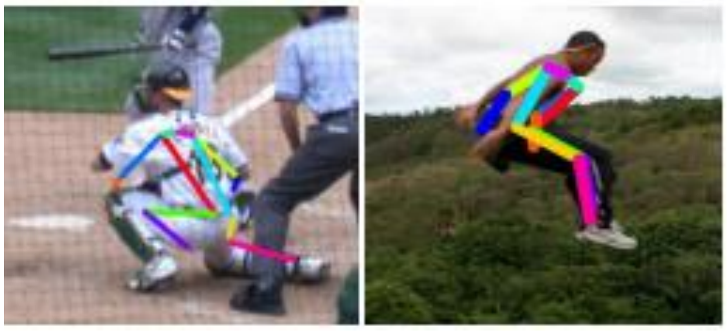


Fig 2.1 : Identifying articulated joints

In Figure 2.1. besides extreme variability in articulations there are many of the joints which are barely visible. Guessing the location of the right arm in the left image is only possible because the rest of the pose anticipate the motion or activity of the person. Similarly, the left body half of the person on the right is not visible at all. These are examples of the need for holistic reasoning. DNNs can naturally provide such type of reasoning. Due these reasons this system uses DNN.

2.1 Deep Learning Model for Pose Estimation:

The proposed system used the following notation. To express a pose, encode the locations of all k body joints in pose vector defined as y = (. . . , y T i , . . .) T , i ∈ {1, . . . , k}, where yi contains the x and y coordinates of the i th joint. A labeled image is denoted by (x, y) where x stands for the image data and y is the ground truth pose vector. Further, since the joint coordinates are in absolute image coordinates, it proves beneficial to normalize them w. r. t. a box b bounding the human body or parts of it. In a trivial case, the box can denote the full image. Such a box is defined by its center bc ∈ as well as width bw and height bh: b = (bc, bw, bh). Then the joint yi can be translated by the box center and scaled by the box size which we refer to as normalization by b:

$$N(\mathbf{y}_i; b) = \begin{pmatrix} 1/b_w & 0 \\ 0 & 1/b_h \end{pmatrix} (\mathbf{y}_i - b_c) \qquad (1)$$

Further, they applied the same normalization to the elements of pose vector N(y; b) = (. . . , N(yi ; b) T , . . .) T resulting in a normalized pose vector. Finally, with a slight abuse of notation, they used N(x; b) to denote a crop of the image x by the bounding box b, which de facto normalizes the image by the box. For brevity we denote by N(·) normalization with b being the full image box.

2.2 Pose Estimation as DNN-based Regression:

In this work, the proposed system treated the problem of pose estimation as regression, where they trained and used a function ψ(x; θ) ∈ R 2k which for an image x regresses to a normalized pose vector, where θ denotes the parameters of the model. Thus, using the normalization transformation from Eq. (1) the pose prediction y ∗ in absolute image coordinates reads

$$y * = N - 1\ (\psi(N(x); \theta))\ (2)$$

Despite its simple formulation, the power and complexity of the method is in ψ, which is based on a convolutional Deep Neural Network (DNN). Such a convolutional network consists of several layers – each being a linear transformation followed by a non-linear one. The first layer takes as input an image of predefined size and has a size equal to the number of pixels times three color channels. The last layer outputs the target values of the regression, in their case 2k joint coordinates.

The proposed system of pose estimation using deepPose is based on Deep Neural Networks (DNNs). The pose estimation is formulated as a DNN-based regression problem towards body joints. This system proposed a cascade of such DNN regressors to result in high precision pose estimates. The approach has the advantage of reasoning about pose in a holistic fashion and has a simple but yet powerful formulation which capitalizes on recent advances in Deep Learning.

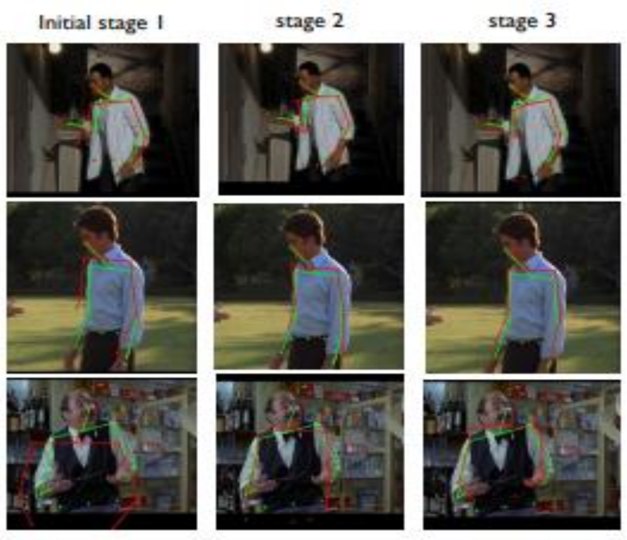


Fig 2.2 : Cascade of Pose Regressors

This system under study uses Cascade of Pose Regressors wherein the pose formulation from the previous section has the advantage that the joint estimation is based on the full image and thus relies on context. However, due to its fixed input size of 220 × 220, the network has limited capacity to look at detail – it learns filters capturing pose properties at coarse scale. These are necessary to estimate rough pose but insufficient to always precisely localize the body joints. In order to achieve better precision, this system proposed to train a cascade of pose regressors. At the first stage, the cascade started off by estimating an initial pose as outlined in the previous section. At subsequent stages, additional DNN regressors are trained to predict a displacement of the joint locations from previous stage to the true location. Thus, each subsequent stage can be thought of as a refinement of the currently predicted pose, as shown in Fig. 2.2. Further, each subsequent stage uses the predicted joint locations to focus on the relevant parts of the image – subimages are cropped around the predicted joint location from previous stage and the pose displacement regressor for this joint is applied on this sub-image. In this way, subsequent pose regressors see higher resolution images and thus learn features for finer scales which ultimately leads to higher precision. Their system uses the same network architecture for all stages of the cascade but learn different network parameters. For stage $s \in \{1, \ldots, S\}$ of total S cascade stages, we denote by θs the learned network parameters.

On comparison of the other methods with this proposed methodology of pose estimation , this methodology gave the most accurate results due to use of cascade of pose regressors.

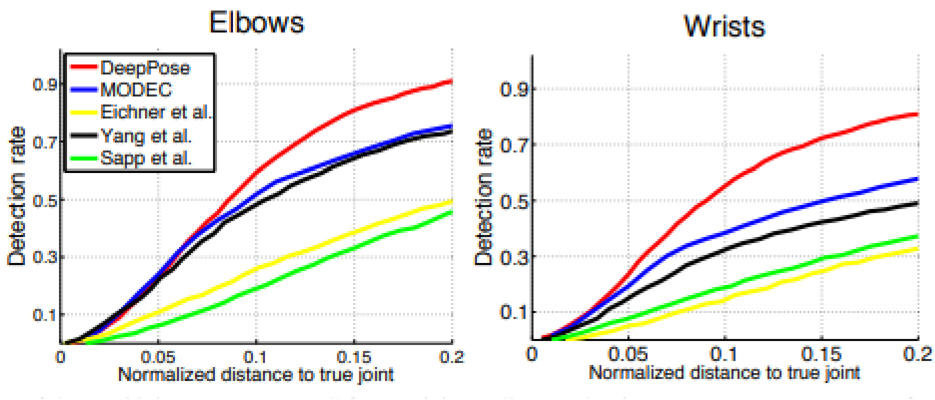


Fig. 2.3 : DeepPose vs other approaches

This graph compares DeepPose, after two cascade stages, with four other approaches and proves that DeepPose and use of cascade stages is one of the best methods of pose estimation.Inspired by the crux of this paper which is pose estimation the system MatchPose is proposed. MatchPose aims to perform pose estimation using PoseNet. The advantage of PoseNet is that it requires minimal specialized hardware and/or cameras, and anyone with a decent. Also it is able to estimate poses in real time which is the main aim of our system.

Chapter 3

Problem Statement

3.1 Drawback of Current Approaches

Traditional way of matching any desired pose involves help from a third person. The third person then judges if the person does the pose correctly or not. However, the precision of the judgement of the third person varies and cannot be trusted. If the user is trying to replicate a yoga or dance pose it is utmost important for the users position to be exactly the same to the actual pose. As of now there is no such system which in real time is able to track the users movements and compare it with the desired pose and help the user replicate the pose. This is where our system MatchPose comes to play.

3.2 Solution To Above Problem

Today learning a skill like dance or yoga requires a lot of commitment and putting in tremendous amount of time and money. However, in this technology driven world it is the need of the hour to use technology to its best. Imagine if you want to learn a particular pose, and replicate it, and doing it is just a click away. This is what this system MatchPose assures. This system provides the user with a web-based interface wherein the user can use an existing pose from provided dataset that he/she wants to replicate. Real time input feed taken through the webcam will be analyzed by the system and compared with the desired pose to give accuracy of user pose. The proposed system will help a user to match whichever pose is desired and in the desired time too. This system intends to make learning poses easy and flexible as per user’s requirements. Thus, MatchPose will match the users pose with desired pose in real time and also provides suggestions to the user to help the user reach the desired pose.

Chapter 4

Project Description

4.1 Overview of Project

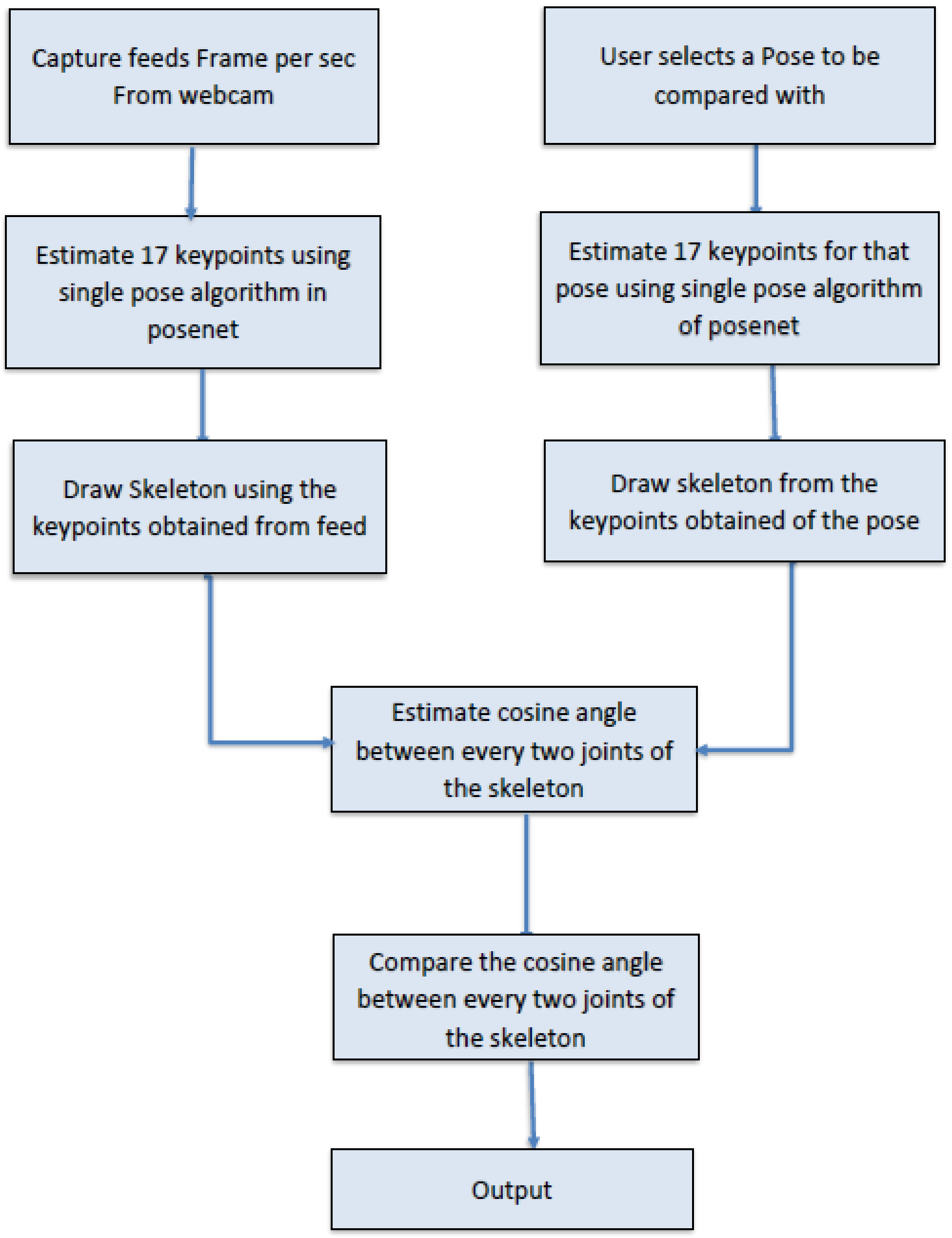
The system compares in real time the user pose taken as input through the webcam with the pose the user wants to replicate. For this purpose we provide the user with a website.

The main page of the website just shows details of our system along with our contact details, also the user can mail to us for further queries.

The next page is a gallery page which contains a set of images from which the user can select any one pose that he wants to replicate. The user will be shown the instructions required to understand how our system works. After this the webcam begins when permission is granted by the user. The user can see the pose image on the side of the webcam input and try to replicate it. The pose skeleton remains black till the pose is not matched. As soon as the pose matches skeleton turns green. Also there is a prompt in the side that tells the user the angle the user needs to make for reaching the desired pose and the angle he is currently making.Thus the user can refer it and accordingly change his pose. This is the overview of MatchPose.

4.2 Architecture Diagram

The architecture diagram of the proposed system is as shown below:



Capture feeds Frame per sec From webcam
User selects a Pose to be compared with
Estimate 17 keypoints using single pose algorithm in posenet
Estimate 17 keypoints for that pose using single pose algorithm of posenet
Draw Skeleton using the keypoints obtained from feed
Draw skeleton from the keypoints obtained of the pose
Estimate cosine angle between every two joints of the skeleton
Compare the cosine angle between every two joints of the skeleton
Output

Fig. 4.2.1: Architecture Diagram of Proposed System

4.2.1 Module 1: Capture user image

In this module, the webcam will capture the image of the user doing pose in front of it. The image will be captured in frames per second format. For Match-pose system we have set the capturing rate to 1-26 frames per second. That means this module will be capturing at most 26 images per second and will estimate the pose for each frame in real time.

4.2.2 Module 2: Keypoint Estimation

In this module the 17 key points from the image will be extracted. These keypoints are nothing but all the bone joints of human body and eyes, ears, nose. These keypoints will be returned by the posenet algorithm in the form of an array of x and y co-ordinates of 17 keypoints.

4.2.3 Module 3: Formation of skeleton

In this module, the skeleton will be formed by joining the keypoints. The skelton will be formed to match the user pose with the selected image. Initially when the user pose is not same as the selected pose, the skeleton will appear in black color. As the user pose matches with the selected pose, the skeleton blinks green.

4.2.4 Module 4: Calculation of cosine angles

In this module the cosine angles will be calculated between each two joints of the skeleton. These angels will be used to tell user how much his pose is differing from the desired pose.

4.2.5 Module 5: Compare the angles

In this module the angles calculated in above module will be used to compare user's pose with the desired pose. Before the comparison in this phase the skeleton will remain black.

4.2.6 Module 6: Output

In the output module, if the user's pose matches exactly with the selected pose, the user's skeleton will blink green. If the user is not able to replicate the pose correctly the skeleton will remain black user and inform user what angles are need to be made and what angles the user is making.

4.3 UML Diagrams

4.3.1 UseCase Diagram

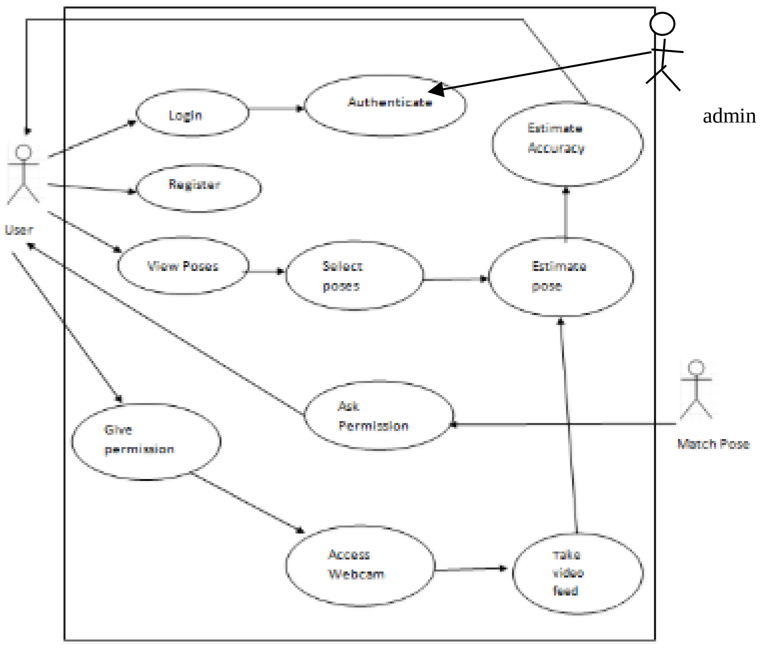


Fig. 4.3.1.1 : Use Case Diagram of the Proposed System

Actors Involved:

1. **User**
2. **MatchPose** (this system)
3. **Admin**

Usecases present:

1. **Register:** User can register itself to customise his preferences with the system
2. **Login:** A user can login into the system to access his stored data and preferences
3. **View Poses:** This functionality lets user to choose from the various available poses in the system.
4. **Ask Permission:** The system will ask permission for accessing his webcam
5. **Give Permission:** The user gives permission for accessing the webcam
6. **Select Poses:** The user chooses the desired pose.
7. **Estimate accuracy:** The system measures the accuracy of the poses made
8. **Estimate pose:** Determines the 17 keypoints of the human

9. **Take video feed:** Capturing video feed from webcam at a rate of frames per sec.

4.3.2 DataFlow Diagram

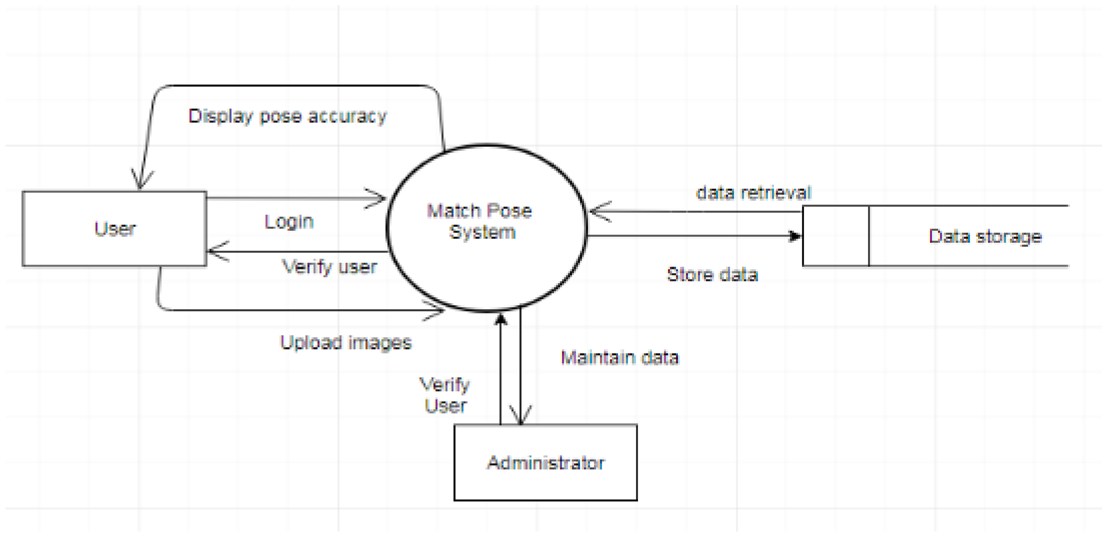


Fig. 4.3.2.1 : DFD-0

DFD level 0:

User: The person who would use the built system

MatchPose: The system which will perform pose estimations and accuracy calculations.

Administrator: The person who would monitor and maintain the system.

Data Storage: The databases present in the system.

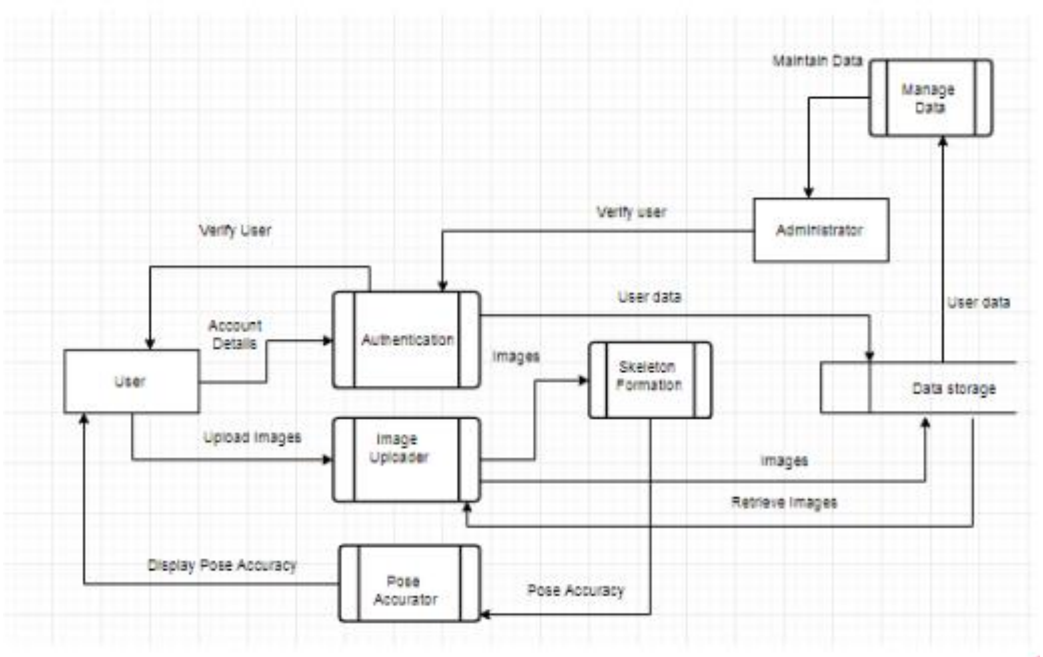


Fig 4.4.2: DFD-1

<u>**DFD Level 1:**</u>

Authentication: Verifies if user is genuine and sends data to data storage.

ImageUploader: Loads image uploaded by the user to the data storage.

ManageData: Takes data from the data storage and provides it to the administrator for maintenance purpose.

SkeletonFormation: Takes the input image/video detects the joints and connects them to form a skeleton.

Pose Accurator: Gives accuracy of the user pose with respect to the desired pose.

Chapter 5

Technologies Used And Requirements

5.1 Technologies Used

5.1.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

5.2 CSS

Cascading Style Sheets, or CSS, are a way to change the look of HTML and XHTML web pages. CSS was designed by the W3C, and is supported well by most modern web browsers. The current version of CSS is CSS3. CSS4 is available, but is split into parts.

One advantage to using CSS is a web page can still be displayed and understood, even if the CSS is not working or removed. CSS code is saved in files with the `.css` file extension.

5.3 JAVASCRIPT

JavaScript often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a programming language that is characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

5.4 PHP

PHP: Hypertext Preprocessor (or simply PHP) is a general-purpose programming language originally designed for web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code may be executed with a command line interface (CLI), embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification.

5.5 XAMPP SERVER

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

XAMPP's ease of deployment means a WAMP or LAMP stack can be installed quickly and simply on an operating system by a developer.

5.2 Hardware And Software Requirement

5.2.1. Hardware Requirement

- Webcam - Minimum 2 MP
- CPU - Minimum i3 generation
- Hard disk - 50 GB
- RAM - 1 GB

5.2.2. Software Requirement

- Windows 7 and above
- Web Technologies - HTML,CSS, JavaScript, PHP
- Server - Xampp server

Chapter 6

Design and Output

6.1 Implementation

6.1.1 Home Page of MatchPose (index.html)

This page has three main parts for the user:

1) **Get Started Part which takes the user to the image set:**

```
<div class="mySlides" width="1000" height="500" >

        <div class="row" width="1000"><div class="column" width="1000">

                        <img  src="img/pose_1.jpg" width="500" height="300" >

        </div><div class="column" width="1000">

        <p style="font-size:25px;"><font color="black"><br>Not sure if you are getting</br><br> the expected
<b>POSE</b>?</br></font></p><br><br><br>

<div class="wrapper">

                        <a href="gallery.php" class="btn-custom1">Get Started!!!</a>

</div></div></div></div>

<div class="mySlides" width="1000" height="500" ><div class="row" width="1000">

<div class="column" width="1000">

                        <img  src="img/danceBall.jpg" width="500" height="300" >

</div>

<div class="column" width="1000">

                        <p style="font-size:25px;"><font  color="black"><br><b>WHY WORRY?</b></br> <br> WE
GOT YOU COVERED </br> <br> USE<b> MATCH POSE </b></br><br></font></p>

                        <a href="gallery.php" class="btn-custom1">Get Started!!!</a>

</div></div></div></div>
```

2) **About Part :**

```
<div class="w3-content w3-section" style="max-width:2000px">

 <div class="section-title">

      <h2> About the System</h2>

</div>

<div class="mySlides" width="1000" height="500" ><div class="row" width="1000"><div class="column"  width="1000">

                <img  src="img/learningPose.jpg" width="500" height="300">

</div>
```

```
<div class="column"  width="1000">

<p style="font-size:25px;"><font  color="black"><br>Are you not satisfied with the</br><br><b> Traditional</b> way of learning?</br></font></p><br><br><br>

                    <a href="gallery.php" class="btn-custom1">Get Started!!!</a>

</div></div></div>
```

3) Contact Part:

```
<form  name="sentMessage" id="contactForm">

<div class="row"><div class="col-md-6"><div class="form-group">

<input id="name" type="text" class="form-control" placeholder="Name" required="required">

<p class="help-block text-danger"></p>

</div></div>

<div class="col-md-6"><div class="form-group">

<input id="email" type="text" class="form-control" placeholder="Email" required="required"><p class="help-block text-danger"></p>

</div></div></div>

<div class="form-group"><textarea name="message" id="message" class="form-control" rows="4" placeholder="Message" required></textarea>

     <p class="help-block text-danger"></p></div>

                    <div id="success"></div>

<input id="submit" onclick="myFunction()" type="button" class="btn btn-custom btn-lg" value="Send Message"></form>
```

6.1.2 Image Gallery (gallery.php)

This page also has three main features

1) Displaying Instruction page on load

```
<div style="top: 50%; left: 50%; display: none;" id="dialog" class="window">

<div id="san">

<a href="#" class="close agree"><img src="close-icon.png" width="25" style="float:right; margin-right: -25px; margin-top: -20px;"></a>

<img src="ins2.JPG" width="450">

</div>
```

2) Fetching all the images from the image folder

```
<?php

    $dirname = "images/";

    $images = glob($dirname."*.jpg");

    foreach($images as $image) {?>
    <li><button onClick="selectImage('<?php echo($image) ?>')"><img src='<?php echo($image)?>'  width="350" height="350" /><br /></button></li>

<?php

        }

    ?>
```

3) Displaying these images in a systematic grid format

```
li{

    list-style-type:none;

    margin-right:50px;

    margin-bottom:30px;

    float:left;

                box-shadow: 10px 10px 5px grey;

}
```

6.1.3 Displaying webcam output and user prompt (camera.html)

```
<title>PoseNet - Camera Feed Demo</title>

    <style>

    .footer {

        position: fixed;

        left: 0;

        bottom: 0;

        width: 100%;

        color: black;

    }

    .footer-text {

        max-width: 600px;

        text-align: center;

        margin: auto;

    }

    @media only screen and (max-width: 600px) {

      .footer-text, .dg {

        display: none;

      }

    }

                #output{

                position: absolute;
```

```
}

#LeftArm{

position:relative;

left:10px;

top:20px;

float:right;

border: 3px solid green;

background-color: yellow;

}

#RightArm{

position:relative;

left:317px;

top:75px;

float:right;

border: 3px solid green;

background-color: yellow;

}

#LeftLeg{

position:relative;

left:624px;

top:131px;

float:right;

border: 3px solid green;

background-color: yellow;

}

#RightLeg{

position:relative;

left:931px;

top:187px;

float:right;

border: 3px solid green;

background-color: yellow;

}

#cat1{

position :relative;
```

```
                left:720px;
                }
                #poster{
                position :relative;
                left:20px;
                top:720px;
                float: bottom;
                }
    </style>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body onload="start()">
<div id="info" style='display:none'>
    </div>
    <div id="loading">
        Loading the model...
    </div>
    <div id='main' style='display:none'>
        <video id="video" playsinline style=" -moz-transform: scaleX(-1);
        -o-transform: scaleX(-1);
        -webkit-transform: scaleX(-1);
        transform: scaleX(-1);
                            display:none
        ">
        </video><canvas id="output"/>
    </div>
                <canvas id="LeftArm" width="300" height="50"></canvas>
                <canvas id="RightArm" width="300"height="50"></canvas>
                <canvas id="LeftLeg" width="300"height="50"></canvas>
                <canvas id="RightLeg" width="300"height="50"></canvas>
                <img id='cat1' src="height="300" width="250" >
                <img id='cat' style="display:none">
                <br>
                <canvas id="poster" height="500" width="500"></canvas>
    <script src="https://cdn.jsdelivr.net/npm/dat.gui@0.7.2/build/dat.gui.js"></script>
    <script src="https://unpkg.com/@tensorflow/tfjs@0.15.0"></script>
    <script src="https://unpkg.com/@tensorflow-models/posenet@0.2.3"></script>
```

```
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
   <script src="PoseCompare.js"></script>
  <script src="demo_util_webcam.js"></script>
   <script src="stats.js"></script>
   <script src="camera.js"></script>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
 <script src="ImageEst.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.js"></script>
<script>
function start() {
  var img=localStorage.imageSrc;
  document.getElementById('cat').src = img;
  document.getElementById('cat1').src = img;
}
</script></body></html>
```

6.1.4 Javascript Page associated with camera.html (camera.js)

```
var maxVideoSize = 513;//513;
var canvasSize = 700;
var stats = new Stats();
var Wpose;
function isAndroid() {
  return /Android/i.test(navigator.userAgent);}
function isiOS() {
  return /iPhone|iPad|iPod/i.test(navigator.userAgent);}
function isMobile() {
  return isAndroid() || isiOS();
}
async function setupCamera() {
  var video = document.getElementById('video');
  video.width = maxVideoSize;
  video.height = maxVideoSize;
  if (navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
    var mobile = isMobile();
    const stream = await navigator.mediaDevices.getUserMedia({
      'audio': false,
      'video': {
        facingMode: 'user',
        width: mobile ? undefined : maxVideoSize,
        height: mobile ? undefined: maxVideoSize}
    });
    video.srcObject = stream;
    return new Promise(resolve => {
      video.onloadedmetadata = () => {
        resolve(video);
      };
    });
  } else {
    var errorMessage = "This browser does not support video capture, or this device does not have a camera";
    alert(errorMessage);
    return Promise.reject(errorMessage);
  }}
async function loadVideo() {
  var video = await setupCamera();
  video.play();

  return video;
}
var guiState = {
  algorithm: 'single-pose',
  input: {
    mobileNetArchitecture: isMobile() ? '0.50' : '0.75',
    outputStride: 16,
    imageScaleFactor: 0.5,
  },
  singlePoseDetection: {
    minPoseConfidence: 0.1,
    minPartConfidence: 0.5,
  },
  multiPoseDetection: {
    maxPoseDetections: 5,
    minPoseConfidence: 0.2,
    minPartConfidence: 0.15,
    nmsRadius: 30.0,
  },
  output: {
    showVideo: true,
    showSkeleton: true,
    showPoints: true,
  },
  net: null,
};
function setupGui(cameras, net) {
  guiState.net = net;
  if (cameras.length > 0) {
    guiState.camera = cameras[0].deviceId;
  }
  var cameraOptions = cameras.reduce((result, { label, deviceId }) => {
    result[label] = deviceId;
    return result;
  }, {});
  var gui = new dat.GUI({ width: 300 });
  var algorithmController = gui.add(
    guiState, 'algorithm', ['single-pose', 'multi-pose']);
  let input = gui.addFolder('Input');
  var architectureController =
    input.add(guiState.input, 'mobileNetArchitecture', ['1.01', '1.00', '0.75', '0.50']);
  input.add(guiState.input, 'outputStride', [8, 16, 32]);
  input.add(guiState.input, 'imageScaleFactor').min(0.2).max(1.0);
  input.open();
  let single = gui.addFolder('Single Pose Detection');
  single.add(guiState.singlePoseDetection, 'minPoseConfidence', 0.0, 1.0);
  single.add(guiState.singlePoseDetection, 'minPartConfidence', 0.0, 1.0);
  single.open();

  let multi = gui.addFolder('Multi Pose Detection');
  multi.add(
    guiState.multiPoseDetection, 'maxPoseDetections').min(1).max(20).step(1);
  multi.add(guiState.multiPoseDetection, 'minPoseConfidence', 0.0, 1.0);
  multi.add(guiState.multiPoseDetection, 'minPartConfidence', 0.0, 1.0);
  // nms Radius: controls the minimum distance between poses that are returned
  // defaults to 20, which is probably fine for most use cases
  multi.add(guiState.multiPoseDetection, 'nmsRadius').min(0.0).max(40.0);
  let output = gui.addFolder('Output');
  output.add(guiState.output, 'showVideo');
  output.add(guiState.output, 'showSkeleton');
  output.add(guiState.output, 'showPoints');
  output.open();
  architectureController.onChange(function (architecture) {
    guiState.changeToArchitecture = architecture;
  });
  algorithmController.onChange(function (value) {
    switch (guiState.algorithm) {
      case 'single-pose':
        multi.close();
        single.open();
        break;
      case 'multi-pose':
        single.close();
        multi.open();
        break;
    }
  });
}
function setupFPS() {
  stats.showPanel(0); // 0: fps, 1: ms, 2: mb, 3+: custom
  document.body.appendChild(stats.dom);
}
function detectPoseInRealTime(video, net) {
  var canvas = document.getElementById('output');
  var ctx = canvas.getContext('2d');
  var flipHorizontal = true; // since images are being fed from a webcam
  canvas.width = canvasSize;
  canvas.height = maxVideoSize;
  async function poseDetectionFrame() {
    if (guiState.changeToArchitecture) {
      // Important to purge variables and free up GPU memory
      guiState.net.dispose();
      // Load the PoseNet model weights for either the 0.50, 0.75, 1.00, or 1.01 version
      guiState.net = await posenet.load(Number(guiState.changeToArchitecture));

      guiState.changeToArchitecture = null;
    }

    // Begin monitoring code for frames per second
    stats.begin();

    // Scale an image down to a certain factor. Too large of an image will slow down
    // the GPU
    var imageScaleFactor = guiState.input.imageScaleFactor;
    var outputStride = Number(guiState.input.outputStride);
                let poses= []
    let minPoseConfidence;
    let minPartConfidence;
    switch (guiState.algorithm) {
      case 'single-pose':
        var pose = await guiState.net.estimateSinglePose(video, imageScaleFactor, flipHorizontal, outputStride);

                            //WebPose(pose);
                            poses.push(pose);
                            //console.log("keypoints from webcam");
                            //console.log(poses);
        minPoseConfidence = Number(
          guiState.singlePoseDetection.minPoseConfidence);
        minPartConfidence = Number(
          guiState.singlePoseDetection.minPartConfidence);
        break;
      case 'multi-pose':
        poses = await guiState.net.estimateMultiplePoses(video, imageScaleFactor, flipHorizontal, outputStride,
          guiState.multiPoseDetection.maxPoseDetections,
          guiState.multiPoseDetection.minPartConfidence,
          guiState.multiPoseDetection.nmsRadius);

        minPoseConfidence = Number(guiState.multiPoseDetection.minPoseConfidence);
        minPartConfidence = Number(guiState.multiPoseDetection.minPartConfidence);
```

```
        break;
      }

      ctx.clearRect(0, 0, canvasSize, canvasSize);

      if (guiState.output.showVideo) {
        ctx.save();
        ctx.scale(-1, 1);
        ctx.translate(-canvasSize, 0);
        ctx.drawImage(video, 0, 0, canvasSize, canvasSize);
        ctx.restore();
      }

      var scale = canvasSize / video.width;

      // For each pose (i.e. person) detected in an image, loop through the poses
      // and draw the resulting skeleton and keypoints if over certain confidence
      // scores
      poses.forEach(({ score, keypoints }) => {
        if (score >= minPoseConfidence) {
          if (guiState.output.showPoints) {
            drawKeypointss(keypoints, minPartConfidence, ctx, scale);
          }
          if (guiState.output.showSkeleton) {
            drawSkeletons(keypoints, minPartConfidence, ctx, scale);
          }
        }
      });

      // End monitoring code for frames per second
      stats.end();

      requestAnimationFrame(poseDetectionFrame);

    }  poseDetectionFrame();}
async function bindPage() {
  // Load the PoseNet model weights for version 1.01
  var net = await posenet.load();

  document.getElementById('loading').style.display = 'none';
  document.getElementById('main').style.display = 'block';

  let video;

  try {
    video = await loadVideo();
  } catch(e) {
    console.error(e);
    return;
  }

  setupGui([], net);
  setupFPS();
  detectPoseInRealTime(video, net);
}
function WebPose(){

		return Wpose;
}
navigator.getUserMedia = navigator.getUserMedia ||
  navigator.webkitGetUserMedia ||
  navigator.mozGetUserMedia;
bindPage(); // kick off the demo
```

6.1.5 Code for Estimating Image keypoints and plotting its skeleton (ImgEst.js)

```
var colors = 'red';
var lineWidths = 5;
var scales=1;
function drawKeypoints(keypoints, minConfidence, ctx, scale = 1) {
  for (let i = 0; i < keypoints.length; i++) {
    var keypoint = keypoints[i];
    if (keypoint.score < minConfidence) {
      continue;}
    var { y, x } = keypoint.position;

    ctx.beginPath();

    ctx.arc(x * scale, y * scale, 3, 0, 2 * Math.PI);

    ctx.fillStyle = color;

    ctx.fill(); }}

async function start(){

var imageScaleFactor = 0.8;

var flipHorizontal = false;

var outputStride = 8;

var maxPoseDetections = 2;

var minPoseConfidence=0.2;

var minPartConfidence=0.15;

var imageElement = document.getElementById('cat');

var net =  await posenet.load();

var canvas = document.getElementById('poster');

var ctx = canvas.getContext('2d')

var poses = await net.estimateMultiplePoses(imageElement, imageScaleFactor, flipHorizontal, outputStride,maxPoseDetections,
minPartConfidence);

 poses.forEach(({ score, keypoints }) => {

  if (score >= minPoseConfidence) {

  drawKeypoints(keypoints, minPartConfidence, ctx, scales);

   draw(keypoints,ctx);

   } });}

  function toTuple({ y, x }) {

  return [y, x];}

  function drawSegmentss([ay, ax], [by, bx], colors, scales, ctx) {

  ctx.beginPath();

  ctx.moveTo(ax * scales, ay * scales);

  ctx.lineTo(bx * scales, by * scales);

  ctx.lineWidth = lineWidths;

  ctx.strokeStyle = colors;

  ctx.stroke();

}

function draw(keypoints,ctx){

setKeypoints(keypoints);

//left elbow joint start

drawSegmentss(toTuple(keypoints[5].position),

      toTuple(keypoints[7].position), colors, scales, ctx);
```

```
drawSegmentss(toTuple(keypoints[7].position),
      toTuple(keypoints[9].position), colors, scales, ctx);
// left elbow joint ends
//right elbow joints
drawSegmentss(toTuple(keypoints[6].position),
      toTuple(keypoints[8].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[8].position),
      toTuple(keypoints[10].position), colors, scales, ctx);
 //right elbow joints ends.
//right knee starts
drawSegmentss(toTuple(keypoints[12].position),
      toTuple(keypoints[14].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[14].position),
      toTuple(keypoints[16].position), colors, scales, ctx);
//right knee ends
//left knee starts
drawSegmentss(toTuple(keypoints[11].position),
      toTuple(keypoints[13].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[13].position),
      toTuple(keypoints[15].position), colors, scales, ctx);
//left knee ends
drawSegmentss(toTuple(keypoints[5].position),
      toTuple(keypoints[6].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[6].position),
      toTuple(keypoints[12].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[11].position),
      toTuple(keypoints[12].position), colors, scales, ctx);
drawSegmentss(toTuple(keypoints[5].position),
      toTuple(keypoints[11].position), colors, scales, ctx);}
start();
```

6.1.6 Code for Calculating angles and comparing poses (PoseCompare.js)

```
var keyPose; //image Pose
var LeftHipDegree,LeftWaistDegree;
var LeftElbowDegree,LeftKneeDegree;
var RightHipDegree,RightWaistDegree;
var RightElbowDegree,RightKneeDegree;
var towards;
var towardss;
var towardsleg;
var towardssleg;
function setKeypoints(pose)
{
	keyPose=pose;
	console.log(keyPose);
	console.log(keyPose[5].position);
	LeftArmAngle();
	RightArmAngle();
	LeftLegAngle();
	RightLegAngle();
}
function getAngle(a,b,c)
{
	var ax=a.x;
	var ay=a.y;
	var bx=b.x;
	var by=b.y;
	var cx=c.x;
	var cy=c.y;
var p12 = Math.sqrt(Math.pow((ax - bx),2) + Math.pow((ay - by),2));
var p13 = Math.sqrt(Math.pow((ax - cx),2) + Math.pow((ay - cy),2));
var p23 = Math.sqrt(Math.pow((bx - cx),2) + Math.pow((by - cy),2));
var resultDegree = Math.acos(((Math.pow(p23, 2)) + (Math.pow(p12, 2)) - (Math.pow(p13, 2))) / (2 * p23 * p12)) * 180 / Math.PI;
return resultDegree;
}
function LeftLegAngle(){
LeftKneeDegree=getAngle(keyPose[11].position,keyPose[13].position,keyPose[15].position);
LeftWaistDegree=getAngle(keyPose[13].position,keyPose[11].position,keyPose[12].position);
if(keyPose[11].position.x> keyPose[15].position.x)
{towardsleg="right";}
else
{towardsleg="left";}
}
function RightLegAngle(){
RightKneeDegree=getAngle(keyPose[12].position,keyPose[14].position,keyPose[16].position);
```

```
RightWaistDegree=getAngle(keyPose[14].position,keyPose[12].position,keyPose[11].position);
if(keyPose[12].position.x> keyPose[16].position.x)
{towardssleg="right";}
else
{towardssleg="left";}
}
function compareLeftLeg(a,b,c,d){ //  (11,13,15,12)
var resultDegree = getAngle(a,b,c);
var resultDegrees = getAngle(b,a,d);
var canvas = document.getElementById('LeftLeg');
 var ctx = canvas.getContext('2d');
 ctx.clearRect(0,0,canvas.width,canvas.height);
 ctx.restore();
 ctx.font = "15px Verdana black";
 ctx.fillText("Left Knee-> "+"Target:"+Math.round(LeftKneeDegree)+" Actual:"+Math.round(resultDegree), 10,10 );
 ctx.fillText("Left Hip-> "+"Target:"+Math.round(LeftWaistDegree)+" Actual:"+Math.round(resultDegrees), 10,25 );
 if( resultDegree>=LeftKneeDegree-10 && resultDegree<=LeftKneeDegree+10){
        if(resultDegrees>=LeftWaistDegree-10 && resultDegrees<=LeftWaistDegree+10)
        {       if(towardsleg=="right")
                {
                        if(a.x>=c.x)
                                return 1;
                        else
                                return 0;
                }
                else
                {
                        if(a.x<c.x)
                                return 1;
                        else
                                return 0;
                }
        }
        else{ return 0;}
}
else{
```

```
                return 0;
}}
function compareRightLeg(a,b,c,d){
var resultDegree = getAngle(a,b,c);
var resultDegrees = getAngle(b,a,d);

var canvas = document.getElementById('RightLeg');
 var ctx = canvas.getContext('2d');
 ctx.clearRect(0,0,canvas.width,canvas.height);
 ctx.restore();
 ctx.font = "15px Verdana black";
 ctx.fillText("Right Knee-> "+"Target:"+Math.round(RightKneeDegree)+" Actual:"+Math.round(resultDegree), 10,10 );
 ctx.fillText("Right Hip-> "+"Target:"+Math.round(RightWaistDegree)+" Actual:"+Math.round(resultDegrees), 10,25 );
 if( resultDegree>=RightKneeDegree-10 && resultDegree<=RightKneeDegree+10){
                                if(resultDegrees>=RightWaistDegree-10 && resultDegrees<=RightWaistDegree+10)
                {
                                if(towardssleg=="left")
                                {                               if(a.x<=c.x)
                                                                return 1;
                                                else
                                                                return 0;
                                                }
                                else
                                {
                                                if(a.x>c.x)
                                                                return 1;
                                                else
                                                                return 0;
                                }
                }
                else{ return 0;}
}
else{
                return 0;
}
```

```
}
function LeftArmAngle(){
LeftElbowDegree=getAngle(keyPose[5].position,keyPose[7].position,keyPose[9].position);
LeftHipDegree=getAngle(keyPose[7].position,keyPose[5].position,keyPose[11].position);
if(keyPose[5].position.x> keyPose[9].position.x)
{towards="right";}
else
{towards="left";}
}
function compareLeftElbow(a,b,c,d){ //  (5,7,9,11)          5-7,7-9 and 7-5,5-11
var resultDegree = getAngle(a,b,c);
var resultDegrees = getAngle(b,a,d);
var canvas = document.getElementById('LeftArm');
 var ctx = canvas.getContext('2d');
 ctx.clearRect(0,0,canvas.width,canvas.height);
 ctx.restore();
 ctx.font = "15px Verdana black";
 ctx.fillText("Left Elbow-> "+"Target:"+Math.round(LeftElbowDegree)+" Actual:"+Math.round(resultDegree), 10,10 );
 ctx.fillText("Left Shoulder-> "+"Target:"+Math.round(LeftHipDegree)+" Actual:"+Math.round(resultDegrees), 10,25 );
if( resultDegree>=LeftElbowDegree-10 && resultDegree<=LeftElbowDegree+10){
        if(resultDegrees>=LeftHipDegree-10 && resultDegrees<=LeftHipDegree+10)
        {
                if(towards=="right")
                {               if(a.x>=c.x)
                                return 1;
                        else
                                return 0;

                }
                else
                {       if(a.x<c.x)
                                return 1;
                        else
                                return 0;
                }
        }
        else{ return 0;}
```

```
                }
else{
                return 0;
}}
function compareRightElbow(a,b,c,d){ //  (6,8,10,12)          6-8,8-10 and 8-6,6-12
var resultDegree = getAngle(a,b,c);
var resultDegrees = getAngle(b,a,d);
var canvas = document.getElementById('RightArm');
 var ctx = canvas.getContext('2d');
 ctx.clearRect(0,0,canvas.width,canvas.height);
 ctx.restore();
 ctx.font = "15px Verdana black";
 ctx.fillText("Right Elbow-> "+"Target:"+Math.round(RightElbowDegree)+" Actual:"+Math.round(resultDegree), 10,10 );
 ctx.fillText("Right Shoulder-> "+"Target:"+Math.round(RightHipDegree)+" Actual:"+Math.round(resultDegrees), 10,25 );
 if( resultDegree>=RightElbowDegree-10 && resultDegree<=RightElbowDegree+10){
                if(resultDegrees>=RightHipDegree-10 && resultDegrees<=RightHipDegree+10)
                {               if(towardss=="left")
                                {               if(a.x<=c.x)
                                                                return 1;
                                                else
                                                                return 0;}
                                else
                                {                               if(a.x>c.x)
                                                                return 1;
                                                else
                                                                return 0;}}
                else{ return 0;}}
else{return 0;}
}
function RightArmAngle(){
RightElbowDegree=getAngle(keyPose[6].position,keyPose[8].position,keyPose[10].position);
RightHipDegree=getAngle(keyPose[8].position,keyPose[6].position,keyPose[12].position);
if(keyPose[6].position.x> keyPose[10].position.x)
{towardss="right";}
else
{towardss="left";}}
```

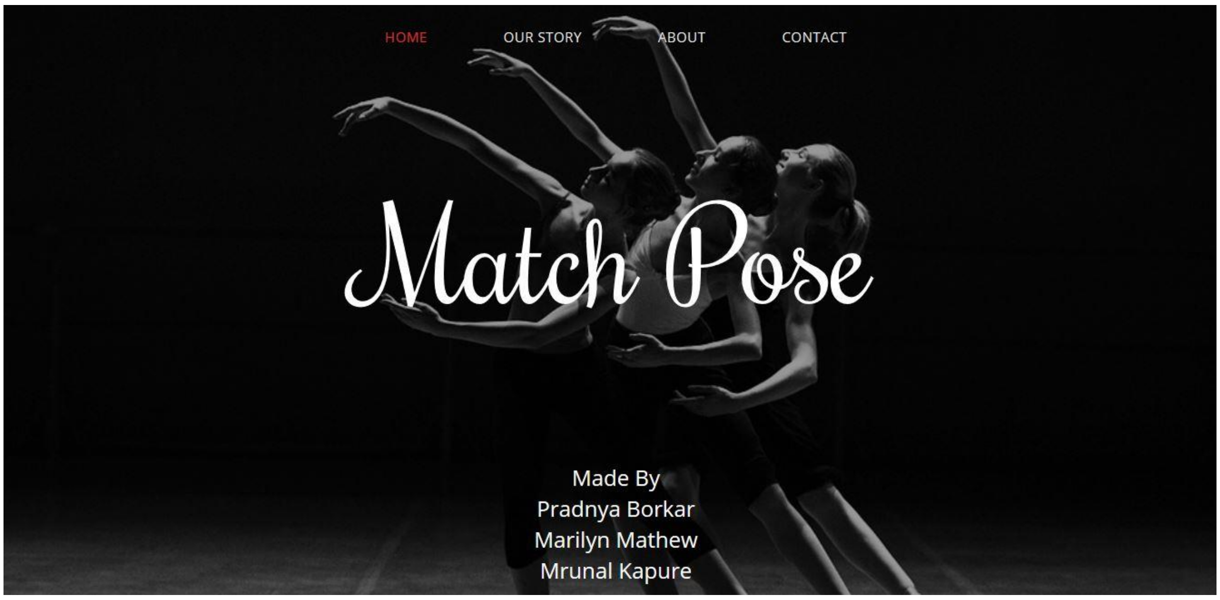
6.1.7 Code for changing the skeleton colour to green in case of pose match (demo_util_webcam.js)

```
var color = 'red';

var lineWidth = 5;

var scale=1;

function toTuples({ y, x }) {

  return [y, x];

}

function drawSegments([ay, ax], [by, bx], color, scale, ctx) {

  ctx.beginPath();

  ctx.moveTo(ax * scale, ay * scale);

  ctx.lineTo(bx * scale, by * scale);

  ctx.lineWidth = lineWidth;

  ctx.strokeStyle = color;

  ctx.stroke();}

function drawSkeletons(keypoints, minConfidence, ctx, scale = 1) {

        int c=0;

if(compareLeftElbow(keypoints[5].position,keypoints[7].position,keypoints[9].position,keypoints[11].position)==1){

                colors='green';

                document.body.style.backgroundColor = "green";




        }

        else{

                colors='black';

        }

if(compareRightElbow(keypoints[6].position,keypoints[8].position,keypoints[10].position,keypoints[12].position)==1){

                colorR='green';         }

        else{

                colorR='black';

        }

if(compareLeftLeg(keypoints[11].position,keypoints[13].position,keypoints[15].position,keypoints[12].position)==1){

                colorsLL='green';}

        else{

                colorsLL='black';}

if(compareRightLeg(keypoints[12].position,keypoints[14].position,keypoints[16].position,keypoints[11].position)==1){colorsRL='green';}

        else{colorsRL='black';}
```

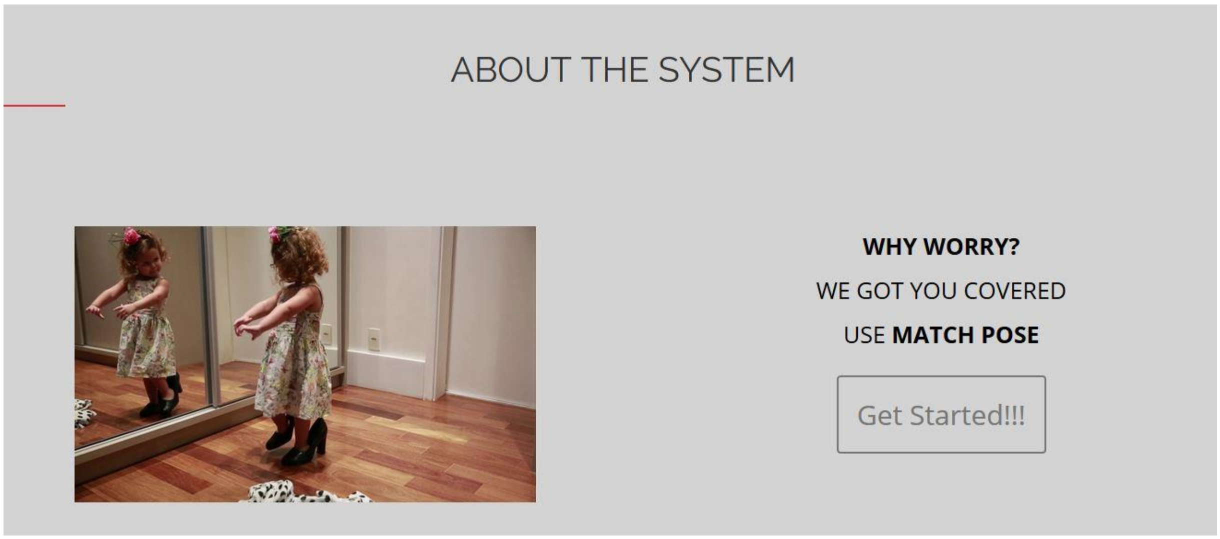
```
if(keypoints[5].score >= minConfidence && keypoints[7].score >= minConfidence){
drawSegments(toTuple(keypoints[5].position),
    toTuple(keypoints[7].position), colors, scale, ctx);
}
if(keypoints[7].score >= minConfidence && keypoints[9].score >= minConfidence){
drawSegments(toTuple(keypoints[7].position),
    toTuple(keypoints[9].position), colors, scale, ctx);}
if(keypoints[6].score >= minConfidence && keypoints[8].score >= minConfidence){
drawSegments(toTuple(keypoints[6].position),
    toTuple(keypoints[8].position), colorR, scale, ctx);
}
if(keypoints[8].score >= minConfidence && keypoints[10].score >= minConfidence){
drawSegments(toTuple(keypoints[8].position),
    toTuple(keypoints[10].position), colorR, scale, ctx);}
if(keypoints[12].score >= minConfidence && keypoints[14].score >= minConfidence){
drawSegments(toTuple(keypoints[12].position),
    toTuple(keypoints[14].position), colorsRL, scale, ctx);
}
if(keypoints[14].score >= minConfidence && keypoints[16].score >= minConfidence){


drawSegments(toTuple(keypoints[14].position),
    toTuple(keypoints[16].position), colorsRL, scale, ctx);
}
if(keypoints[11].score >= minConfidence && keypoints[13].score >= minConfidence){
                        drawSegments(toTuple(keypoints[11].position),
    toTuple(keypoints[13].position), colorsLL, scale, ctx);
}
if(keypoints[13].score >= minConfidence && keypoints[15].score >= minConfidence){


drawSegments(toTuple(keypoints[13].position),
    toTuple(keypoints[15].position), colorsLL, scale, ctx);}
if(keypoints[5].score >= minConfidence && keypoints[6].score > minConfidence){
drawSegments(toTuple(keypoints[5].position),
    toTuple(keypoints[6].position), color, scale, ctx);
}
if(keypoints[6].score >= minConfidence && keypoints[12].score >= minConfidence){
```

```
drawSegments(toTuple(keypoints[6].position),
   toTuple(keypoints[12].position), color, scale, ctx);
}
if(keypoints[11].score >= minConfidence && keypoints[12].score >= minConfidence){

drawSegments(toTuple(keypoints[11].position),
   toTuple(keypoints[12].position), color, scale, ctx);
}
if(keypoints[5].score >= minConfidence && keypoints[11].score >= minConfidence){

drawSegments(toTuple(keypoints[5].position),
   toTuple(keypoints[11].position), color, scale, ctx);
}}
function drawBoundingBoxs(keypoints, ctx) {
  var boundingBox = posenet.getBoundingBox(keypoints);
  ctx.rect(boundingBox.minX, boundingBox.minY,
    boundingBox.maxX - boundingBox.minX, boundingBox.maxY - boundingBox.minY);
  ctx.stroke();
}
sync function renderToCanvass(a, ctx) {
  var [height, width] = a.shape;
  var imageData = new ImageData(width, height);
  var data = await a.data();
  for (let i = 0; i < height * width; ++i) {
    var j = i * 4;
    var k = i * 3;
    imageData.data[j + 0] = data[k + 0];
    imageData.data[j + 1] = data[k + 1];
    imageData.data[j + 2] = data[k + 2];
    imageData.data[j + 3] = 255;
  }
  ctx.putImageData(imageData, 0, 0);
}
function renderImageToCanvass(image, size, canvas) {
  canvas.width = size[0];
  canvas.height = size[1];
  var ctx = canvas.getContext('2d');
  ctx.drawImage(image, 0, 0);
}
function drawHeatMapValuess(heatMapValues, outputStride, canvas) {
  var ctx = canvas.getContext('2d');
  var radius = 5;
  var scaledValues = heatMapValues.mul(tf.scalar(outputStride, 'int32'));
  drawPointss(ctx, scaledValues, radius, color);
}
function drawPointss(ctx, points, radius, color) {
  var data = points.buffer().values;
  for (let i = 0; i < data.length; i += 2) {
    var pointY = data[i];
    var pointX = data[i + 1];
    if (pointX !== 0 && pointY !== 0) {
      ctx.beginPath();
      ctx.arc(pointX, pointY, radius, 0, 2 * Math.PI);
      ctx.fillStyle = color;
      ctx.fill();
    } }}
function drawOffsetVectorss(
  heatMapValues, offsets, outputStride, scale = 1, ctx) {
  var offsetPoints = posenet.singlePose.getOffsetPoints(
    heatMapValues, outputStride, offsets);
  var heatmapData = heatMapValues.buffer().values;
  var offsetPointsData = offsetPoints.buffer().values;
  for (let i = 0; i < heatmapData.length; i += 2) {
    var heatmapY = heatmapData[i] * outputStride;
    var heatmapX = heatmapData[i + 1] * outputStride;
    var offsetPointY = offsetPointsData[i];
    var offsetPointX = offsetPointsData[i + 1];
    drawSegments([heatmapY, heatmapX], [offsetPointY, offsetPointX],
      color, scale, ctx);
  }}
```

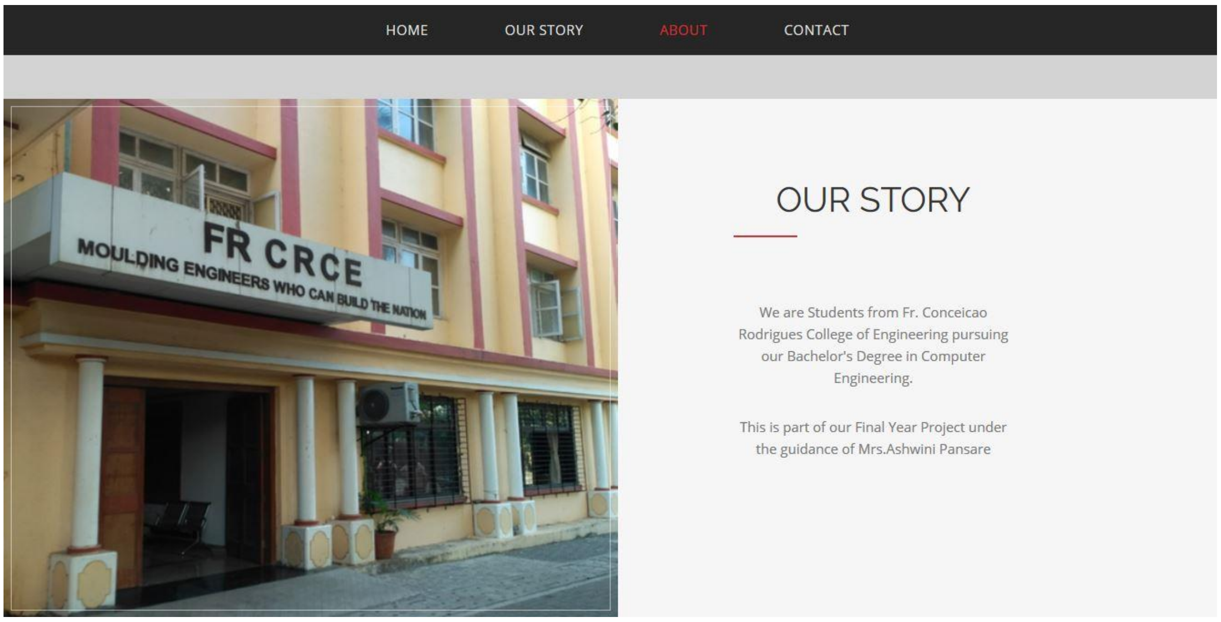
6.2 Snapshots



6.2.1 MatchPose Home Page



6.2.2 About Part on MatchPose Home Page

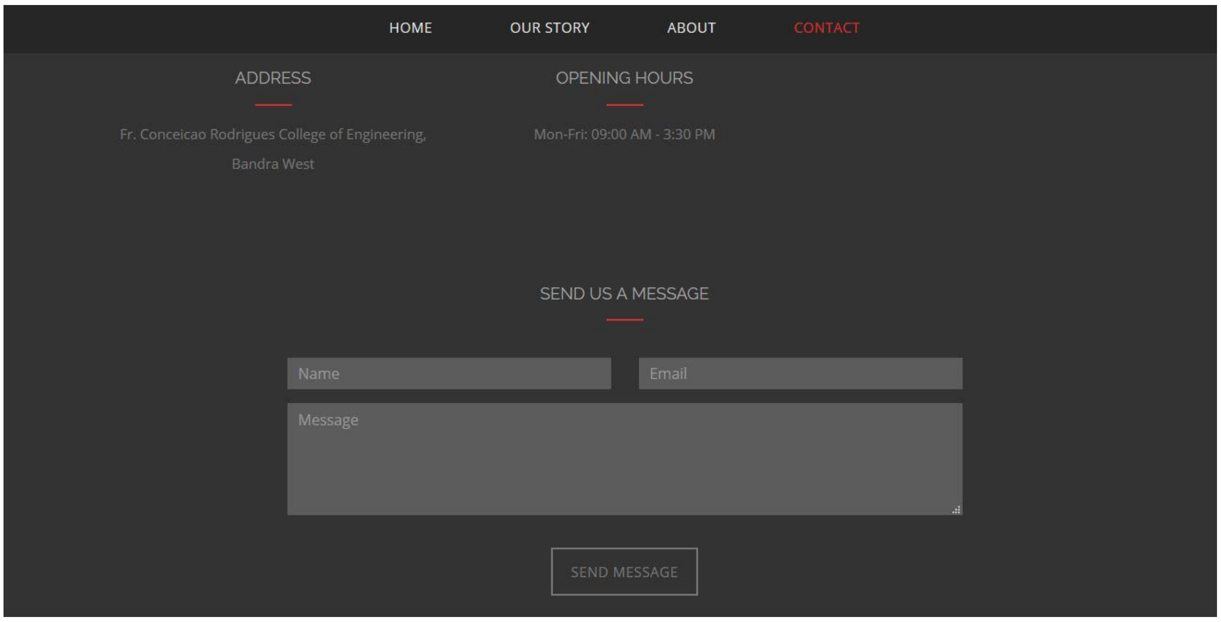


OUR STORY

We are Students from Fr. Conceicao Rodrigues College of Engineering pursuing our Bachelor's Degree in Computer Engineering.

This is part of our Final Year Project under the guidance of Mrs.Ashwini Pansare

6.2.3 Our Story on MatchPose Home Page



ADDRESS

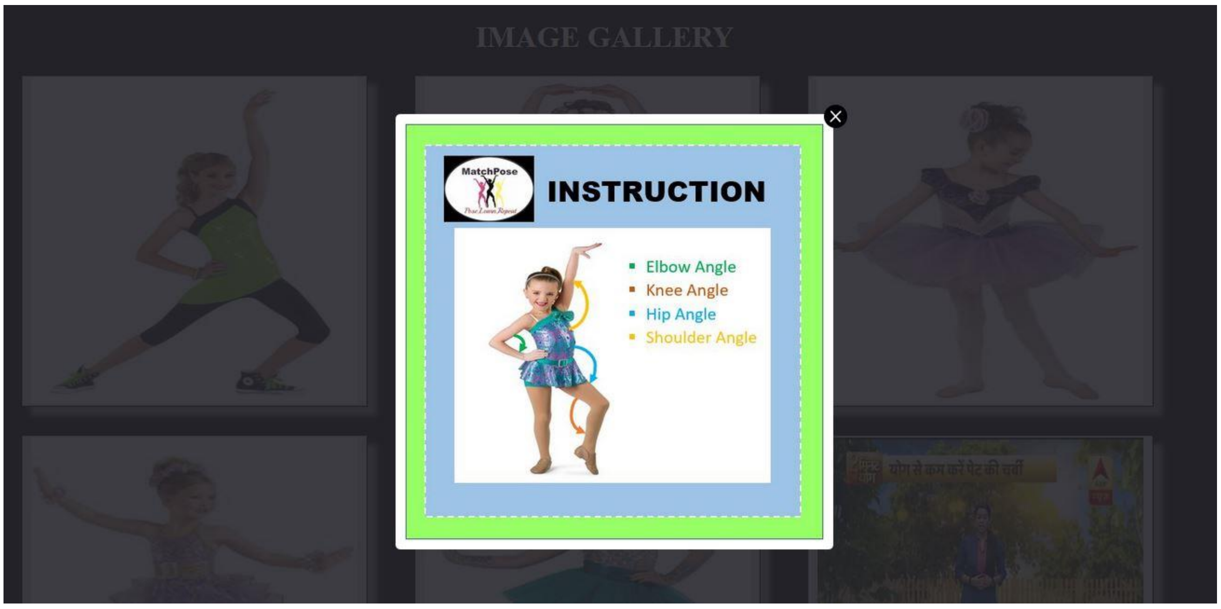
Fr. Conceicao Rodrigues College of Engineering, Bandra West

OPENING HOURS

Mon-Fri: 09:00 AM - 3:30 PM

SEND US A MESSAGE

6.2.4 Contact us Part on MatchPose Home Page



6.2.5 Instructions Displayed before image selection

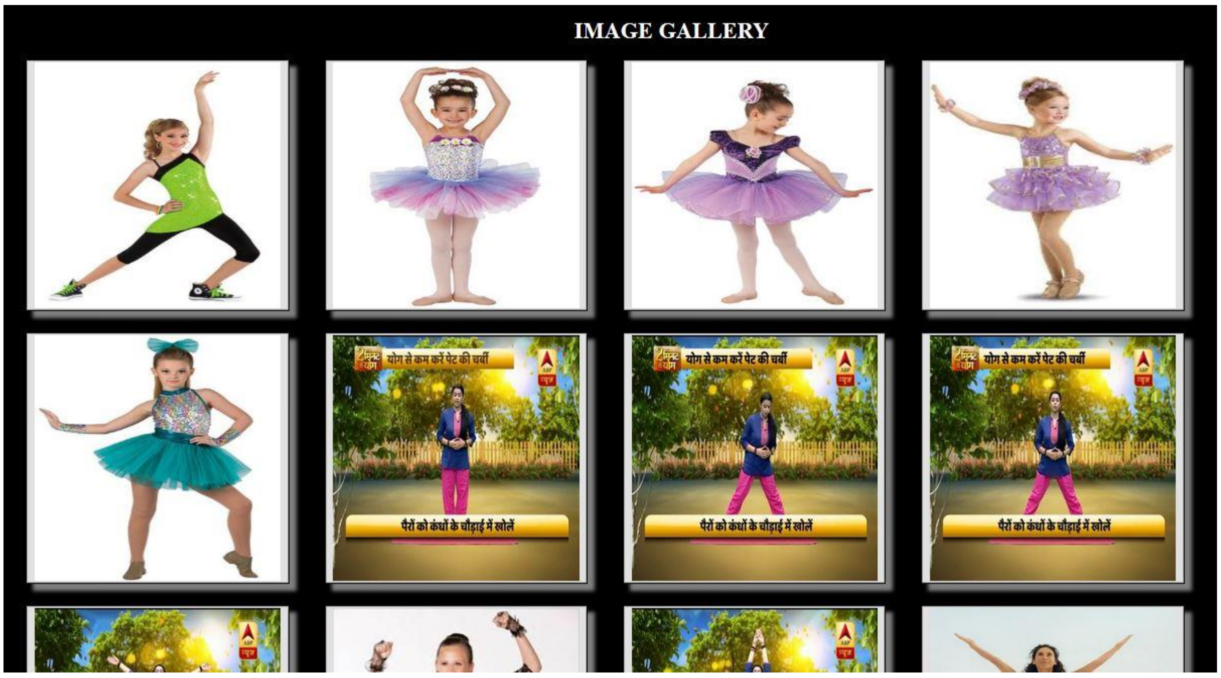
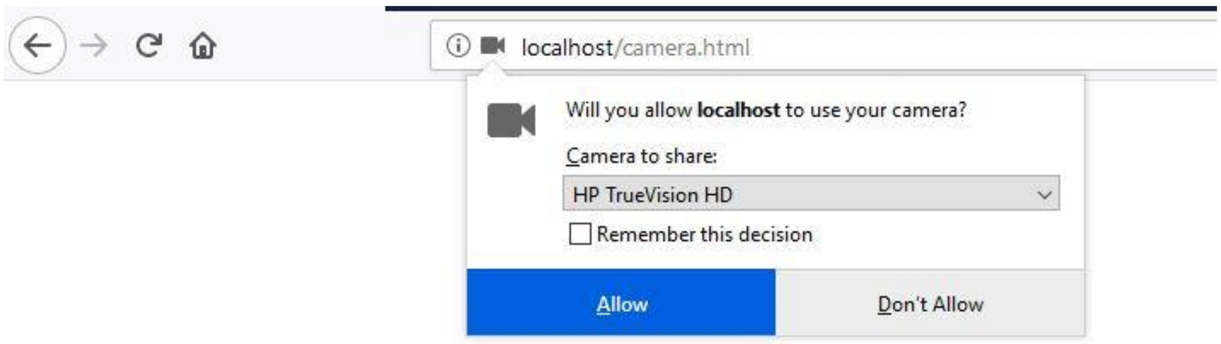


IMAGE GALLERY
योग से कम करें पेट की चर्बी
पैरों को कंधों के चौड़ाई में खोलें

6.2.6 Image Gallery Page



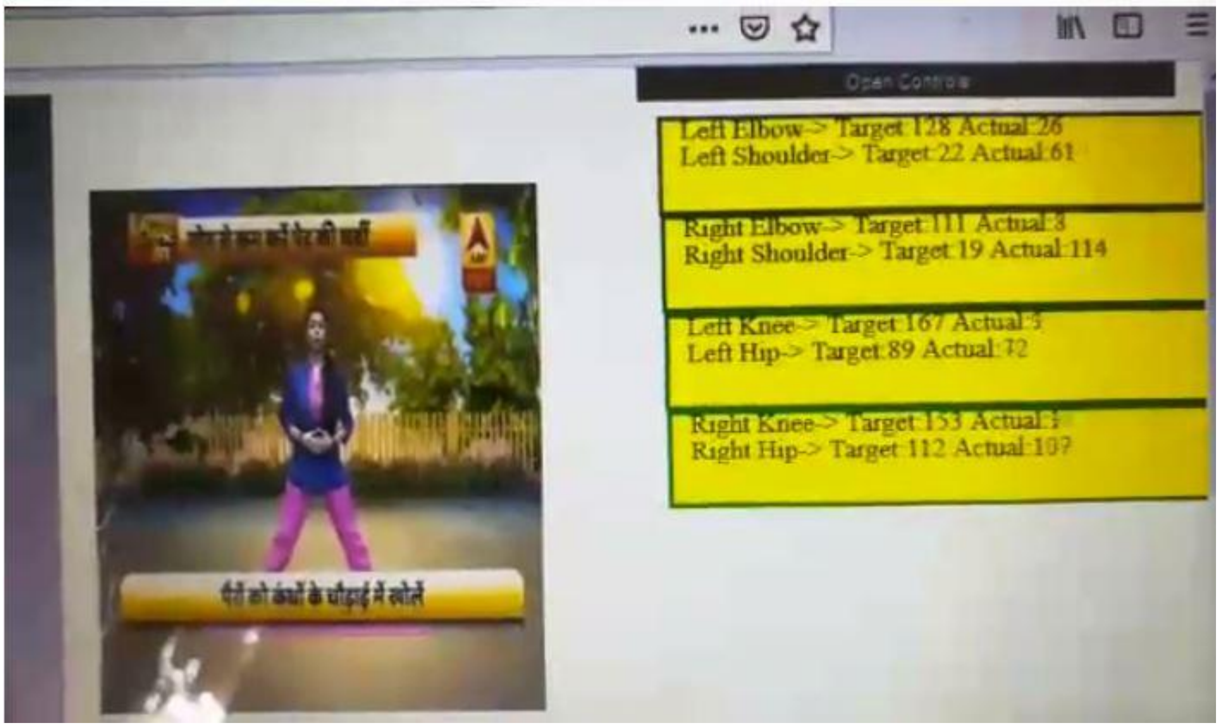
Will you allow **localhost** to use your camera?

Camera to share:

HP TrueVision HD

☐ Remember this decision

6.2.7 WebBrowser asking for permission to access camera



Left Elbow-> Target:128 Actual:26
Left Shoulder-> Target:22 Actual:61

Right Elbow-> Target:111 Actual:3
Right Shoulder-> Target:19 Actual:114

Left Knee-> Target:167 Actual:5
Left Hip-> Target:89 Actual:72

Right Knee-> Target:153 Actual:[illegible]
Right Hip-> Target:112 Actual:107

6.2.8 User Prompt and preview image

Chapter 7

Future Enhancements And Conclusions

7.1 Future Enhancements:

MatchPose system enables users to match a desired pose. Now upgrading this, in future systems can be developed to learn dance or yoga through videos on youtube or other platforms by realtime comparison of the video pose and users pose. This will be a revolution in the field of learning, since it will not just save the users money in investing into classes to learn such skills but also makes learning extremely flexible and fun. This enhancement will give way to the new way of learning where a third person assistance is not required.

7.2 Conclusions:

The system is capable of taking user poses through webcam feed, evaluating the required angles for comparison and comparing the desired pose with user pose in real time. The system also allows the user to select an image from the image gallery provided. The user is guided with a prompt that will help the user to rectify his pose and reach to the desired pose. Thus, MatchPose is able to help the user learn new poses.

References

[1] Alexander Toshev and Christian Szegedy, "*DeepPose: Human Pose Estimation via Deep Neural Networks* ", 2014.

[2] https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5

[3] https://medium.com/tensorflow/move-mirror-an-ai-experiment-with-pose-estimation-in-the-browser-using-tensorflow-js-2f7b769f9b23

[4] Ching-Hang Chen and Deva Ramanan, "*3D Human Pose Estimation = 2D Pose Estimation + Matching*", 2017.

[5] https://becominghuman.ai/single-pose-comparison-a-fun-application-using-human-pose-estimation-part-2-4fd16a8bf0d3

[6] Vivek Maik,Jinho Park, Daehee Kim and Joonki Paik, "*Model-Based Human Pose Estimation and Its Analysis Using Hausdorff Matching*", 2015.

[7] https://beta.observablehq.com/@lorenries/estimating-pose-in-the-browser-with-posenet-and-tensorflow-

[8] Ronald Poppe, Dirk Heylen, Anton Nijholt and Mannes Poel, "*Towards real-time body pose estimation for presenters in meeting environments* ", 2005.

[9] https://beta.observablehq.com/@lorenries/estimating-pose-in-the-browser-with-posenet-and-tensorflow-

[10] https://www.npmjs.com/package/@tensorflow-models/posenet