PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

Posture Recognition and Correction in an Active Training Scenario

Submitted by

Varadaraya G Shenoy (01FB15EEC273)
Vibhav Hosahalli Venkataramaiah (01FB15EEC274)
Vignesh S P (01FB15EEC275)

January - April 2019

under the guidance of

Internal Guide

Mrs. Rajini M
Assistant Professor
Department of Electronics and Communication
PES University
Bengaluru -560085

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

PROGRAM B.TECH

PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

Posture Recognition and Correction in an Active Training Scenario

Submitted by

Varadaraya G Shenoy (01FB15EEC273)
Vibhav Hosahalli Venkataramaiah (01FB15EEC274)
Vignesh S P (01FB15EEC275)

January - April 2019

under the guidance of

Internal Guide

Mrs. Rajini M
Assistant Professor
Department of Electronics and Communication
PES University
Bengaluru -560085

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

PROGRAM B.TECH

CERTIFICATE

This is to certify that the Report entitled

Posture Recognition and Correction in an Active Training Scenario

is a bonafide work carried out by

Varadaraya G Shenoy (01FB15EEC273)

Vibhav Hosahalli Venkataramaiah (01FB15EEC274)

Vignesh S P (01FB15EEC275)

In partial fulfillment for the completion of 8th semester course work in the Program of Study B.Tech in Electronics and Communication Engineering, under rules and regulations of PES University, Bengaluru during the period Jan – Apr. 2019. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature with date & Seal	*Signature with date & Seal*	*Signature with date & Seal*
(Mrs. Rajini M.)	*Dr. Anuradha M*	*Dr. B. K. Keshavan*
Internal Guide	*Chairperson*	*Dean - Faculty of Engg. &Technology*

Name and signature of the examiners:

1.

2.

3.

DECLARATION

We, Varadaraya G Shenoy, Vibhav Hosahalli Venkataramaiah, Vignesh S P**,** hereby declare that the report entitled, "***Posture Recognition and Correction in an Active Training Scenario",*** is an original work done by us under the guidance of **Mrs. Rajini M**, Assistant Professor, ECE Department and is being submitted in partial fulfillment of the requirements for completion of 8th Semester course work in the Program of Study B.Tech in Electronics and Communication Engineering.

PLACE: Bengaluru
DATE:

NAME AND SIGNATURE OF THE CANDIDATES

1.

2.

3.

ABSTRACT

In any sport or weight training session, the posture maintained by the person is most important, not only to ensure that the person performs his best, but also to ensure their safety. This is not usually a concern when the person trains under a trained professional, who ensures that the person is maintaining the correct posture. But in a self-taught session, the person training who in most cases is an amateur may end up severely injuring himself. In this project we aim to develop a platform that checks if the person training is maintaining the correct posture, and in case he is not doing so, provide the corrections necessary.

ACKNOWLEDGEMENTS

We owe our gratitude to our university, PES for providing us a platform to apply the knowledge we have amassed over the duration of our course and work on the project. We would like to sincerely thank Dr Anuradha M., Head of the Department of Electronics and Communication for allowing us to undertake this project.

Our mentor Mrs. Rajini M has been a constant source of guidance and inspiration, for which we are deeply indebted and grateful. Her role has been pivotal in improving our understanding and helping us to approach our project in a logically sound manner.

We appreciate the constructive criticism and helpful inputs by the members of our panel, Dr. Koshy K George and Mrs. Karpagavalli S who have participated in sparking our thought process.

CONTENTS

LISTING OF FIGURES

LISTING OF TABLES

Chapter 1

Introduction

With the increase in information sharing and widespread of the internet, information is available to us at our fingertips at any point of time. This along with the increase in the culture of Do it Yourself (DIY) and self-taught practices has led to people not relying on any outside help for their tasks, be it for studies or development of new skills. Added to this, the recent developments in camera technology, the feature detection of the subject, in this case the person is much easier. In this project we aim to use these features to our advantage to develop a suitable platform to track the user's posture and correct it when necessary.

1.1 Literature Survey

While reviewing the work previously done related to our objective, we came across a couple of applications that currently exists albeit a little primitive.

(1) HomeCourt AI

Developed by the NEX Team Inc., this mobile application build only for Apple OS, uses the camera of the mobile to track, chart and count the basketball shots made by the person in the video in real-time, along with providing instantaneous video review along with its statistical analysis.

By using the mobile phones camera along with image recognition algorithms in the back end, the HomeCourt application is able to measure a vast number of variables such as the trajectory of the shot being made, the position of the body and the height of the jump being made, in order to help the player understand how he/she can improve their form while shooting.

The players can record the shots they've made on their device, and the application will go on track the player and determine what worked with their shot and what didn't, along with identifying where the player is missing and making his/her shot. This application does not track every single action made by the player, but once the player has made a shot, it will track the trajectory of the shot and also track the form of the player, aspects such as where his/her feet are planted. This sort of feedback helps the players to understand the minute adjustments they can make such as release speed and jump height, in order to improve their shooting percentage over time.

While this application is not designed to be robust, the kind used in advanced tracking applications, which are commonly used in highly advanced training facilities, this mobile application provides a simple plug and play platform for getting feedback on the player's progress instantly.

(2) Kaia Health, The Perfect Squat Challenge

Germany-based Kaia Health has launched an AI-powered motion tracking technology called the Perfect Squat Challenge app. The goal being that the app will function as a personal trainer and give real-time feedback on the form of their squats.

Quoting Maximilian Strobel, head of Kaia Health's AI Lab, "In the future, this technology will integrate within our medical device apps for diseases and conditions such as back pain creating a scalable, cost-effective therapeutic tool. This democratizes access to high-quality, bespoke fitness, rehabilitation and physiotherapy - and could reduce the burden on health services."

Users have to keep their smart phone upright either on a table or against a wall. After which they have to move about seven feet away from the screen.

When the user opens the application, they are guided by a virtual trainer, which talks them through, throughout their squat exercise. Using the mobile phone's camera, the application tracks the points of interest in the user's body. People can then review their pose versus the ideal pose for the exercise and adjust as advised. As the user continues the exercises, Kaia also gives audio and video instructions to the user.

(3) Kinect

Microsoft produced Kinect, a range of motion sensing cameras. Originally the Kinect was designed as a gaming accessory for the Microsoft Xbox gaming console and Windows Personal Computers. Designed as a webcam-style accessory, it allows the user to interact with their console without depending on the controller, using gestures and spoken commands.

The detection of humans in a video has been done using Kinect. A single Kinect sensor gives depth map data streams. This data has been used to obtain real-time skeleton based human tracking as seen in [1]. The data available from the Kinect sensor can be used for not only action representation but also for posture recognition. The depth map of the Kinect and other parameters are used to build 3D models and essentially to find the entire body of the person. The various key point parameters are used to obtain a skeletal build which in turn is used in posture recognition.

1.2 Objective

The objective of this project is to develop a posture recognition and correction platform. This platform should be able to detect the current posture of the person of interest in the video and provide the steps necessary for correcting his/her posture, if and when it is improper.

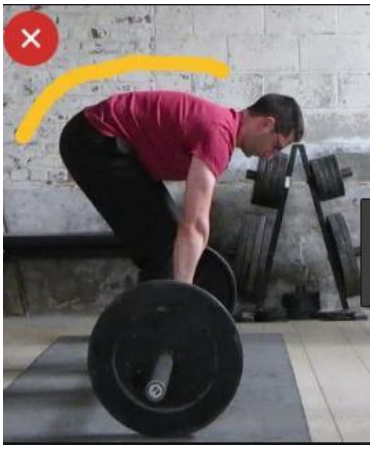
1.3 Motivation

The motivation behind this project is that in any active training scenario such as playing any sport or in a weight training session, the posture of the person is of utmost importance not only to get the required results, but mainly to ensure the person's safety.

This is generally not an issue when a person trains under a coach or a trainer, who ensures that the player is maintaining the required posture. But in a self-taught scenario, without proper guidance, a person, who in most cases an amateur, might severely injure himself and may even end up having some permanent damage, all due to improper posture.

This is where our platform will be useful, giving much needed feedback to ensure that the person maintains the required posture, so that one gets the desired results while ensuring his/her safety.

The figure below illustrates how our objective should ideally work:



(a) Incorrect posture detection*

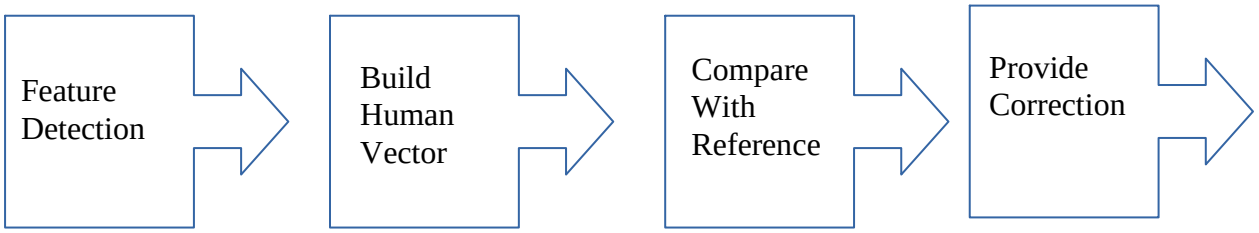
(b) Appropriate Correction Steps*

Fig 1.1 Project Objective

*The Images shown are for representation only.

1.4 Methodology

The process followed to achieve the above-mentioned objective is best depicted in the form of a flow chart.



Let's have a brief look at all the stages before we explain each block in the next few chapters.

1.4.1 Feature Detection

The most important evaluation key points are:

(1) Viewpoint change

(2) Dimensional change

(3) Brightness

Few methods can be used to detect the said features (i.e.) edge detection, corner detection, blob detection. The features could also be evaluated using Feature Descriptors such as SIFT, HOG.

Edge Detection:

It uses different mathematical methods that identify points in an image which has discontinuities.

Corner Detection:

A corner can be defined as an intersection of 2 edges. The superiority of a corner detecting algorithm is determined by how well it can detect the same corner in images which are similar under different conditions.

Blob Detection:

A blob is defined as a region in an image where certain properties are nearly constant. These techniques are mainly used, as they can provide matching information throughout the region, this is not possible with edge detecting and corner detecting algorithms.

For an object in the image, points that are extractable from the image can be used to obtain a "feature descriptor" for that object. This description can be later used as a means of identifying that particular object, in different test cases, where other objects are present as well.

1.4.2 Human Vector Build

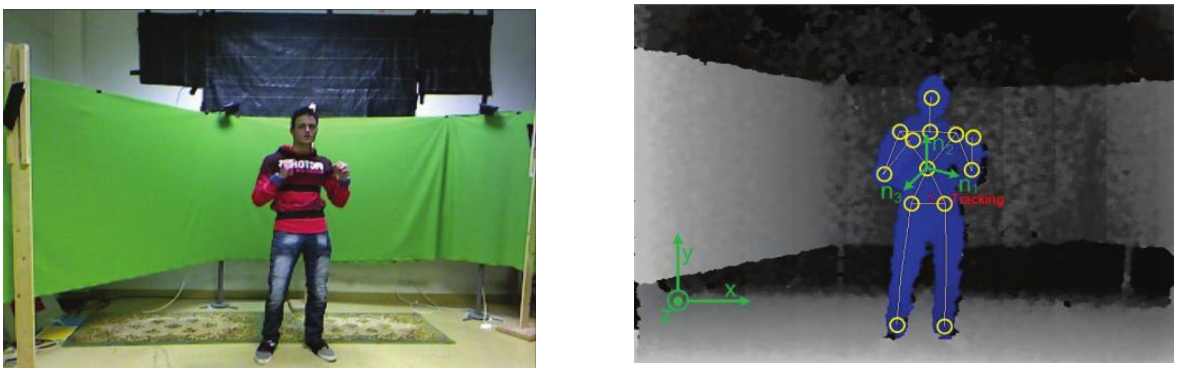
Human pose estimation largely deals with the problem of localizing anatomical key points (i.e.) "parts", it is mainly focused on locating the body parts of individuals in the video/image. The silhouette of the human body on its own is not sufficient to provide us with the key points. We need to have a different representation of the human pose to help us move forward in pose estimation. The technique needed should build a human vector or a skeleton on top of the human body whilst marking of necessary key points.

For this purpose, 2 different approaches are used:

(1) Image Processing Approach: The image is reduced to a skeleton of itself. This action is performed on the silhouette obtained from the previous stage.

(2) Deep learning Approach: Uses datasets which contains the joints important for pose estimation. Further using confidence maps and part-affinity fields, a skeletal build is generated for the respective pose which is used for analysis.

The illustration below gives us an approximate look of how the image processing approach works:



(a) (b)

Fig 1.2 Human Vector Build

1.4.3 Compare with Reference and Correction

The main focus while comparing the test with the reference model is to check the angles between the joints of interest. For this purpose, two methods are most convenient: Hough Transform and co-ordinates method.

Hough Transform: This is a popular technique used in image processing, which is used detect any shape in the image, if that shape can be represented mathematically. It is possible to detect even the slightest distortion present in the shape. This is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the even the slightest distortion of the shape. We use it for a line for our application.

Co-ordinate Method: The classic method of using the co-ordinate axes values to determine the angles of various line segments on the human vector.

After obtaining the angles between various joints, the user can run the video of the required squat and the necessary frames are extracted. These frames have to be the lowest point of the exercise which is where the ideality is compared. The angles obtained from the select frames and then compared with an ideal reference. The user then gets the feedback of whether the posture is a good or a bad one along with necessary angles. For a bad posture, the correction is provided so as to make the pose as ideal as possible.

1.5 Organization of the Dissertation

This project report is organized as follows. Chapter 2 discusses various feature detection algorithms, what its advantages and disadvantages are, and which algorithms are most suitable for platform. In Chapter 3, we discuss the building of the human vector using image processing algorithms along with its pros and cons. In Chapter 4 as well, we discuss building the human vector, but this chapter talks about using a deep learning algorithm. Chapter 5 talks about how comparisons between test and reference video can be made along with how the necessary correction feedback can be given back to user. And finally In Chapter 6, we discuss the scope for future work along with the final discussions.

Chapter 2

Preprocessing and Feature Detection

2.1 Introduction

For an efficient implementation of any feature detection algorithm, preprocessing is a necessary and essential starting point. Preprocessing removes the noise present in the input to make sure that no discontinuities exist which may affect the output.

Feature detection algorithms help in detecting key points that need to be mapped and analyzed carefully to pull out information that is valuable to the problem at hand [2].

Let us now look at how the preprocessing is done and the different algorithms of feature detection used, why they succeed/fail and how it helps us in making further comparisons.

2.2 Video Preprocessing

For pre-processing we use the Gaussian smoothening algorithm to remove any noise present. This algorithm blurs the image by applying the Gaussian function on the image, transforming each pixel in the image.

A typical 2D Gaussian function is given by,

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{2.1}$$

In Eq 2.1, x and y are the pixel coordinates and σ is the standard deviation of the Gaussian distribution.

This acts as a low pass filter, thereby enhancing the image structure at different scales.

2.3 Feature Detection

Let us analyze each feature detection algorithm mentioned in the introduction chapter. For better understanding of the results of each of these algorithms, the same image Fig 2.1 has been fed into each of these algorithms [3].

2.3.1 Edge Detection

This is an image processing algorithm used for the detection of boundaries in an image. These algorithms work by detecting the discontinuities present in the image [4].

We have implemented 4 such algorithms:

(1) Morphology Based Detection

(2) Canny Edge Detection

(3) Haar Wavelet based Edge Detection

(4) Coiflet Wavelet based Edge Detection

2.3.1.1 Morphology Based Detection

Transformations that are based on the image shape are known as Morphological Transformations. These operations are generally performed on images that are binary. Morphology based operations involve 2 inputs, the original image, which is binary and a kernel or structuring element, which is responsible for determining the nature of the operation [4].

Fig 2.1 Input Image

Fig 2.2 Rectangular Kernel

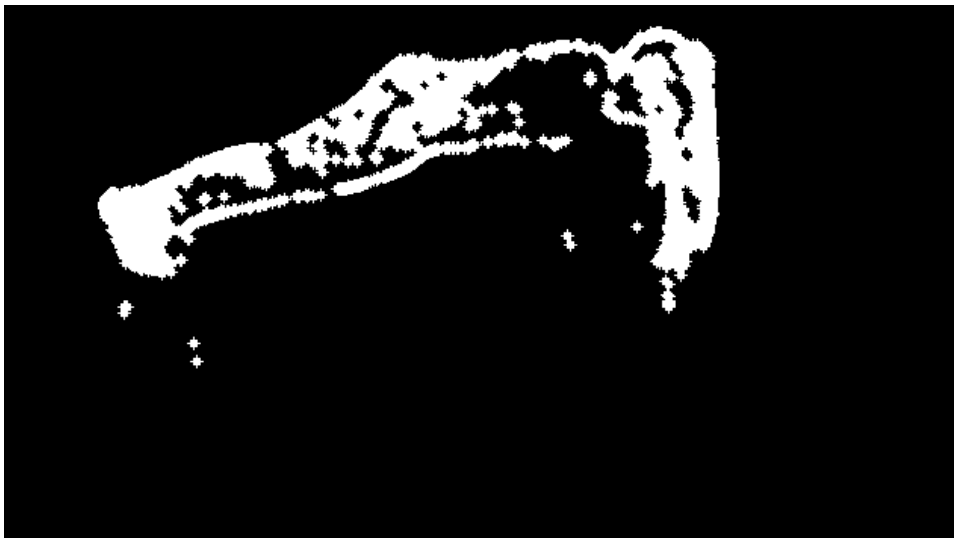


Fig 2.3 Cross Kernel

2.3.1.2 Canny Edge Detection

This operator detects a variety of edges in an image by means of its multi-stage algorithm [4]. By using an edge detection technique, we can extract data regarding the structure of different objects and this helps by decreasing the data to be processed. To meet the requirements of good edge detection algorithm, the edge detector uses, a sum of 4 exponential terms, which can be estimated to the 1st derivative of Gaussian function.

The algorithm can be elaborated as follows:

(1) Noise removal by using Gaussian filtering which smoothens the image. As edges are easily affected by noise and to avoid detection of false edges and loss of edges already present, it is crucial to filter out noise.

(2) Determination of the image's intensity gradients: In order to account for the various directions in which edges may be present, the canny edge detector, 4 filters are used for the detection of edges in all directions of the image. Using edge detection operators such as Perwitt, Robert and Sobel operator, we get the 1st derivative along the x & y direction. The edge gradient and direction is determined as:

$$G=\sqrt{G_x^2+G_y^2} \tag{2.2}$$

$$\Theta=atan\,2\left(G_y^2,G_x^2\right) \tag{2.3}$$

, where G is computed using the hypotenuse function and the arctangent function.

(3) "Non-maximum suppression" is applied to find "the largest" edge. Once the gradient calculation is applied, the edges obtained are still blurred. In order to obtain only one accurate response for an edge, we suppress all gradients except for the local maxima, by applying non-maximum suppression, which will provide coordinates where intensity change is sharpest thereby removing falsely detected edges.

(4) A double threshold is applied to find the most probable edges. Once non-maximum suppression is applied, the edges that remain, give a more precise illustration of the real edges. Nevertheless edges due to color variation and noise still exist.

(5) Suppress edges which are neither strong nor connected to strong edges, and thus finalize the detection of edges. To obtain a more accurate result, edge pixels that are not strong and its neighborhood pixels are analyzed. Given that there is at least one strong edge pixel that is in the blob, the weak edge is preserved.

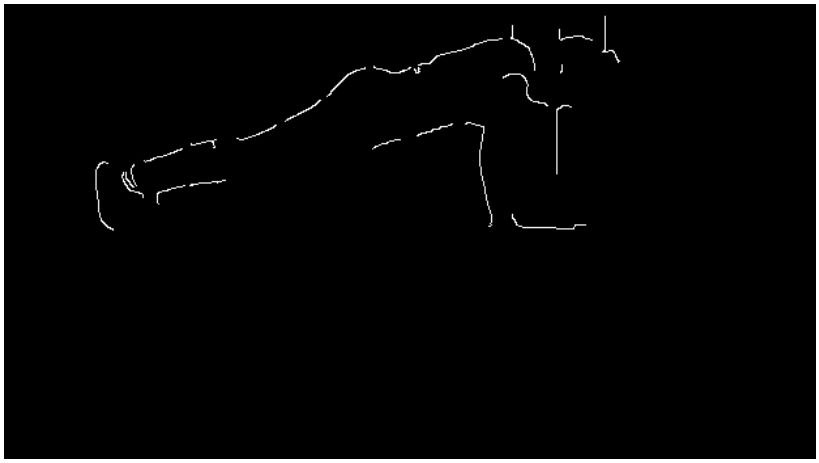


Fig 2.4 Canny Edge

Fig 2.5 Canny Edge with Thresholding

Fig 2.6 Canny Edge with Thresholding and Morphological Operations

2.3.1.3 Haar Wavelet Based Edge Detection

Haar wavelet is a sequence of "rescaled square-shaped functions" which form a wavelet family or basis when put together [5]. Wavelet analysis allows a target function over an interval to be represented as an orthonormal basis.

The Haar wavelet's mother function Ψ (t) is defined as:

$$\Psi(t)=\begin{cases}1, 0\le t<\frac{1}{2}\\ -1, \frac{1}{2}\le t<1\\ 0, otherwise\end{cases} \tag{2.4}$$

Its scaling function:

$$\Psi(t)=\begin{cases}1, 0\le t<1\\ 0, otherwise\end{cases} \tag{2.5}$$

For every integer pair (k, n), the Haar function on the real line is given by:

$$\Psi_{n,k}(t)=2^{\frac{n}{2}}\Psi(2^n t-k) \tag{2.6}$$

, where t belongs to R

The Haar functions are pair wise orthogonal as seen below:

$$\int_{\mathbf{R}} \psi_{n_1,k_1}(t)\psi_{n_2,k_2}(t)\,dt = \delta_{n_1,n_2}\delta_{k_1,k_2}, \tag{2.7}$$

, where δ i, j represents the Kronecker delta.

The Haar wavelet has several notable properties:

"A continuous real function with compact support can be approximated uniformly by linear combinations of {Ψ (t), Ψ (2t), Ψ (4t)... Ψ (2nt)} and their shifted functions. This can be extended to function spaces where the function can be approximated using continuous functions."

"Any continuous real function on [0,1] can also be approximated uniformly on [0,1] by linear combinations of the constant function Ψ(t), Ψ(2t), Ψ(4t),...., Ψ(2nt)and their shifted functions."

Orthogonality in the form,

$$\int_{-\infty}^{\infty} 2^{(n+n_1)/2}\psi(2^n t - k)\psi(2^{n_1} t - k_1)\, dt = \delta_{n,n_1}\delta_{k,k_1}. \tag{2.8}$$

Here δ i, j represents the Kronecker delta. ψ (t) 's dual function is ψ (t) itself.

Wavelet/scaling functions with different scale n have a functional relationship since,

$$\begin{aligned}\varphi(t) &= \varphi(2t) + \varphi(2t-1)\\ \psi(t) &= \varphi(2t) - \varphi(2t-1),\end{aligned} \tag{2.9.1)- (2.9.2}$$

Coefficients of scale n can be determined by using coefficients of scale n+1:

$$\text{If } \chi_w(k,n) = 2^{n/2}\int_{-\infty}^{\infty} x(t)\varphi(2^n t - k)\, dt$$

$$\text{and } \mathrm{X}_w(k,n) = 2^{n/2}\int_{-\infty}^{\infty} x(t)\psi(2^n t - k)\, dt$$

then

$$\begin{aligned}\chi_w(k,n) &= 2^{-1/2}\left(\chi_w(2k,n+1) + \chi_w(2k+1,n+1)\right)\\ \mathrm{X}_w(k,n) &= 2^{-1/2}\left(\chi_w(2k,n+1) - \chi_w(2k+1,n+1)\right).\end{aligned} \tag{2.10.1)- (2.10.2}$$

In DWT (Discrete Wavelet Transform), properties such as time resolution and frequency resolution are realized by using filters (high pass and low pass) and sub-sampling respectively. Scaling is achieved by successively passing the input signal through half-band cut off low and high-pass filters.

For application of DWT in 2D space, it is achieved by first applying the transform to the columns of the image, which will consistently reduce the size of all columns by half, and then it is similarly applied to the rows. The resultant image would be reduced or sub-sampled by 4. Haar wavelets were used here.

Haar wavelets decompose an image of size N×M into four sub-images of sizes N/2×M/2, where N and M are even. The decomposed images are the approximation image (A) and vertical (V), horizontal (H) and diagonal (D) differences' images.

Edges can be found in regions of high contrast; therefore, more attention was given to the coefficients derived from high-pass filtering of the image. Hence, the Haar decomposed images useful for edge detection are the differences' images.

Fig 2.7 Haar Wavelet Based Edge Detection

2.3.1.4 Coiflet Wavelet Based Edge Detection

Coiflet wavelets are discrete wavelets that have "vanishing moments and scaling functions" [5]. One of the most significant properties of Coiflet is that it is nearly symmetric.

Theorem 1: For a wavelet system {φ, φ', Ψ, Ψ', h, h', g, g'}, the following three equations are equivalent:

$$\mathcal{M}_{\tilde{\psi}}(0,l] = 0 \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$\sum_n (-1)^n n^l h[n] = 0 \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$H^{(l)}(\pi) = 0 \qquad \text{for } l = 0, 1, \ldots, L-1; \qquad \text{(2.11.1)- (2.11.3)}$$

One importance of this is that it relates the vanishing moment property of a wavelet into an equivalent property of the corresponding low pass filter (in the time or the frequency domain). The latter can be used directly in the filter design procedure for the wavelet system. This theorem is one of the most essential results in wavelet theory.

Theorem 2: For a wavelet system {φ, φ', Ψ, Ψ', h, h', g, g'}, the following six equations are equivalent:

$$\mathcal{M}_{\phi}(t_0, l] = \delta[l] \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$\mathcal{M}_{\phi}(0, l] = t_0^l \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$\widehat{\phi}^{(l)}(0) = (-jt_0)^l \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$\sum_n (n - t_0)^l h[n] = \delta[l] \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$\sum_n n^l h[n] = t_0^l \qquad \text{for } l = 0, 1, \ldots, L-1$$

$$H^{(l)}(0) = (-jt_0)^l \qquad \text{for } l = 0, 1, \ldots, L-1 \qquad \text{(2.12.1)- (2.12.6)}$$

And similar equivalence holds good for φ' and h'.

Coiflet wavelets have a large computational overhead and uses higher window overlap. 6 scaling and wavelet function coefficients are used. A smoother wavelet is obtained by increasing pixel averaging and differencing. This filter follows the structure as that of both "Haar and Daubechies".

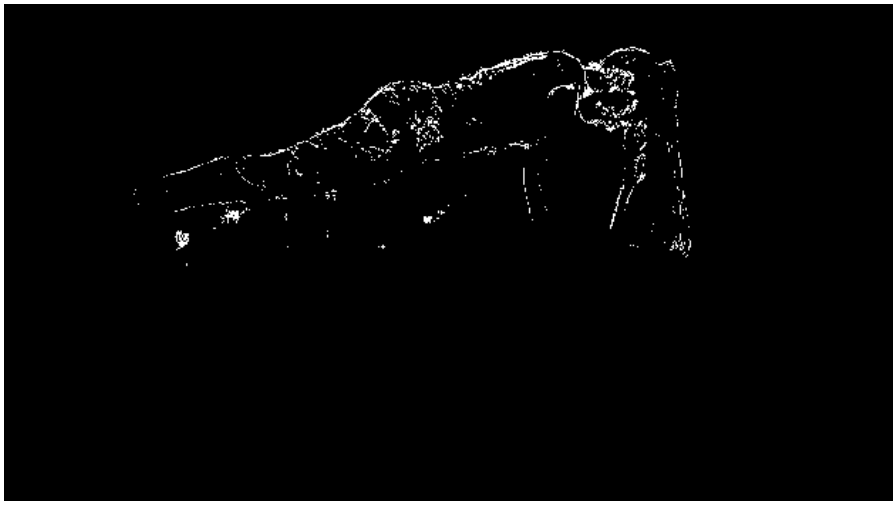


Fig 2.8 Coiflet Wavelet Based Edge Detection

2.3.2 Connected Component Labeling

"Connected-component labeling" (CCL) is an algorithm derived from graph theory, where subsets of connected components are labeled distinctively as per the rules given [5]. CCL is used to discover areas which are connected in both binary and color images.

Using the input data, a graph is constructed which consists of vertices and connecting edges. The vertices hold necessary information for comparison, while the edges signify connected "neighbors". Based on the connectivity of the vertex and its neighboring values, an algorithm labels them while traversing the graph. These image graphs establish the connectivity, which can be either a "4-connected or 8-connected" neighborhood. Once labeling is completed, the graph can be partitioned into subsets, such that the original data can be obtained back and analyzed.

One component at a time:

This is a simple and fast technique to implement and understand. It is based on graph traversal methods in graph theory. In short, as the first connected component pixel is found, all the pixels connected to that pixel are labeled before moving on to the next pixel in the image.

To achieve this, a linked list is formed which keeps the interconnected pixel indexes. The defining of linked list makes use of "a depth or a breadth first search". In this process, the use of different strategies has no change in the end result. The easiest kind of a "last in first out queue" fed as a singly linked list will result in a "depth first search strategy".

The input is binary and the pixels either belong to the background or foreground, and pixels in the foreground which are connected components are the ones that are preferred. CCL follows the following steps:

(1) Begin from the first image pixel and set current label to 1.

(2) If the current pixel is not labeled and is a foreground pixel, allocate the current label to the pixel and append it in a queue, then go to (3). If the current pixel is a labeled or a background pixel, then repeat (2) for the next pixel.

(3) Remove an element from the queue, and examine its neighbors. If a neighbor is a non-labeled foreground pixel, assign the current label to it and append it to the queue. Repeat (3) until the queue is empty.

(4) Return back to step (2) for the next pixel and add 1 to the current label.

Two-pass:

This algorithm repeats through the 2D binary data. 2 passes are made over the image. In the 1st pass, a provisional label is allocated and the uniformities are noted, and in the 2nd pass, the provisional label is substituted by a label that is the smallest in its equivalence class.

The neighbor pixels' labels are checked for connectivity. 8-connectivity uses the top-right, top, top left and left neighbors while the 4-connectivity makes use of only the current pixel's top and left neighbors. To decide the label value assigned to the current pixel certain conditions are looked into:

"(1) If the left pixel has the same value as the current pixel, they share a common space. Allocate the same label to the current pixel. If not, then check next condition.

(2) If the pixels top and left of the current pixel have the same value but a different label as the current pixel, the 2 neighboring belong to the same region and are most likely merged. Assign the current pixel the minimum of the 2 labels, and record their uniformity. If not, then check next condition.

(3) If the left pixel has a different value and the top pixel has the same value as the current pixel, assign the top pixel's label to the current pixel. If not, then check next condition.

(4) If the pixels top and left of the current pixel have different pixel values than current pixel, then create a new label and assign a new pixel to it."

The procedure is continued by the algorithm and crafts new region labels as and when it is essential. The main factor determining the speed of the algorithm is how the merging is performed. Using the "union-find data structure", this algorithm offers exceptional performance in tracking the equivalence relationships.

Perform raster scanning through each and every data element repetitively as part of the 1st pass.

If the background does not contain the element,

(a) Obtain the current element's neighboring elements.

(b) If neighbors are absent, label the current element in a unique manner and proceed.

(c) If not, then assign the smallest label among the current element's neighbors.

(d) Record the neighboring labels' uniformity.

Perform raster scanning through each and every data element repetitively as part of the 2nd pass if it is a background element.

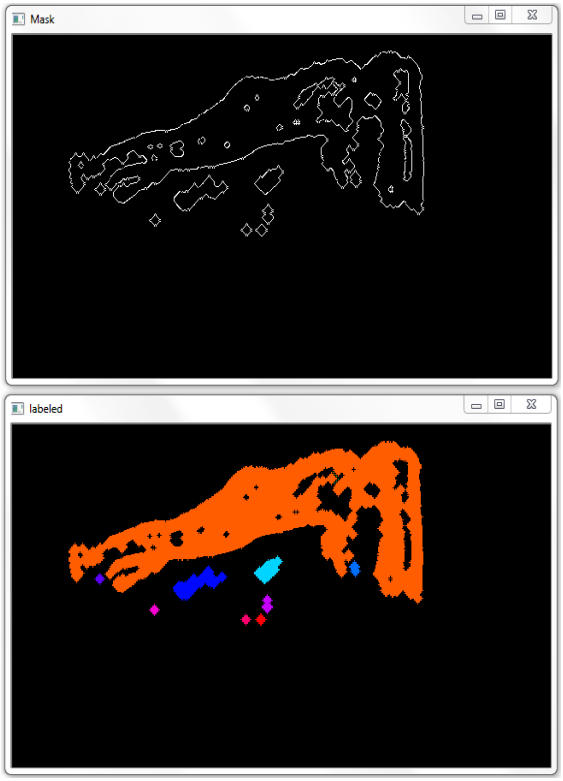


Fig 2.9 CCL with Canny Edge

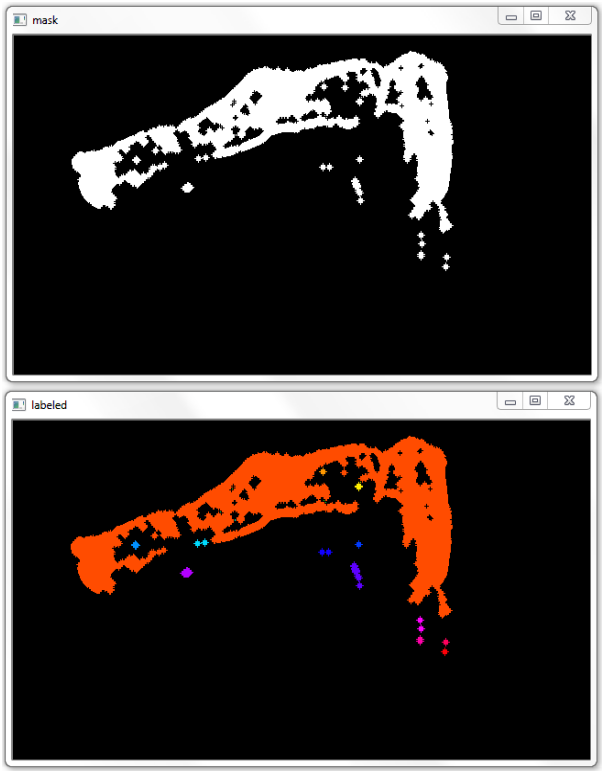


Fig 2.10 CCL with morphological operations

2.3.3 Corner Detection

Corner Detection is a technique utilized in computer vision to deduce the image contents. We have implemented one corner detection algorithm, namely Harris Corner Detector [6].

2.3.3.1 Harris Corner Detector

A corner is the intersection of two edges (sudden change in image brightness). Corners are important image features that are scale invariant, rotation invariant and illumination invariant. Even though corners consist of only a small percentage of the image, they contain the most essential features of image information, and they can be utilized to reduce the amount of processed information for various computer vision areas.

"Harris Corner Detector" is a corner detection operator that is generally used to detect corners and deduce features of an image. It was established due to the progress of Moravec's corner detector.

Harris corner detector algorithm operates in five major steps:

(1) Convert image from color to grayscale.

(2) Compute spatial derivatives along x and y direction.

(3) “Structure tensor setup”, also known as “the second-moment matrix”, is a matrix obtained from the gradient of a function. This matrix sums up the major gradient directions in the specified region for a particular point and the measure of coherence of those directions.

(4) Harris response calculation, done by computing the smallest Eigen value of the structure tensor.

(5) “Non-maximum suppression”, to identify the optimal values for corner indication, the local maxima is found as corners within the window which is a 3x3 filter.

Fig 2.11 Harris Corner Detection

2.3.4 Blob Detection

Blob detection refers to the concept of identifying points and/or image regions that vary in color or brightness with respect to its surroundings.

We have implemented 3 such algorithms:

(1) Histogram of oriented Gradients

(2) Background Subtraction

(3) GrabCut algorithm

2.3.4.1 Histogram of Oriented Gradients

The HOG ("Histogram of Oriented Gradients") is a feature description algorithm used for object detection in image processing [4]. This algorithm counts the number of gradient orientations occurring in local blocks in the image. This algorithm is analogous to edge detection histograms, shape contexts and SIFT, but the difference is that the computations are performed on a dense network of evenly spaced cells and it uses "overlapping local contrast normalization" to get better accuracy.

The concept of HOG descriptor is that the shape and appearance of an object in an image can be illustrated using the intensity gradients' distribution or the edges' direction. The image is split into small regions connected to each other called cells, and the HOG directions for pixels inside each cell is calculated. The descriptor is the combination of these histograms. In order to better the accuracy, the contrast of the local histograms can be normalized by determining an intensity measure across a large region called a block, and then making use of this measure to normalize all cells inside the block. This normalization leads to "better invariance to shadowing and illumination changes".

The primary advantage of the HOG descriptor over other descriptors is that as the descriptor works on local cells, it is geometrically and photo metrically invariant, except for the object orientation. These kinds of variations would only show up in the larger spatial regions. By the use of coarse spatial sampling, strong photometric normalization and fine orientation sampling, the individual body part movement of people can be overlooked given that an upright position is maintained. Therefore, the HOG descriptor is mostly suited for human detection in images.

This algorithm consists of 5 steps:

(1) "Gradient computation". The general method is to apply the "1-D cantered, point discrete derivative mask" horizontally, vertically or both. Specifically, this method calls for filtering the intensity or color data of the image with filter kernels [-1, 0, 1] and [-1, 0, 1].

(2) "Orientation binning" creates the cell histograms. Within each cell, every pixel generates a weighted vote for the histogram channel based on orientation and the values obtained during computation of the gradient. The cells themselves can either be radial or rectangular in shape, and the channels of histogram are spread evenly, depending on whether the gradient is "unsigned" or "signed". In tests, the magnitude of the gradient usually gives the best results.

(3) “Descriptor Blocks”, to take into consideration the changes in the image such as contrast and brightness, the strength of the gradients must be normalized locally; this required the grouping of cells together to form larger areas which are spatially connected. The final HOG descriptor is obtained by concatenating the elements of all the normalized cell histograms from each block region. Each cell makes more than 1 contribution to the final descriptor since the blocks typically overlap.

(4) “Block Normalization”, which changes the range of values to a more suitable one.

(5) “Object Recognition”, HOG descriptors are used to detect objects by feeding them to a “machine learning algorithm” as features.

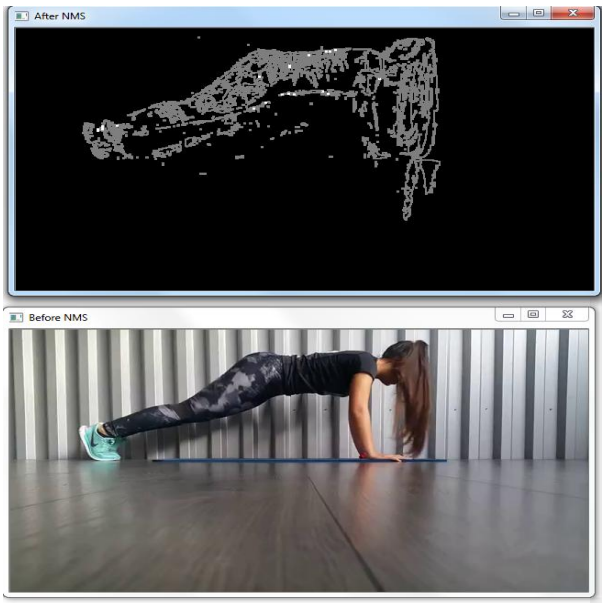


Fig 2.12 HOG Feature Descriptor

2.3.4.2 Background Subtraction

Background subtraction is an important pre-processing step in many image processing applications. This is used in applications where we need to extract the objects alone. The objective is to distinguish a mobile foreground from a stationary background [7].

If a single image with the static background is available, the task is easy, subtract the new image from the background, which will explicitly provide us the foreground images. But in most cases, these types of images may be unavailable, so the background has to be extracted from the limited image resources available. The presence of shadows makes the task all the more complicated. Since the shadows of the objects is also mobile, straightforward subtraction will falsely mark the shadow also as part of the foreground which is undesirable.

MOG:

This algorithm is based on a "Gaussian Mixture-based Background/Foreground Segmentation". This algorithm models each background pixel in the image using a combination of K (3 to 5) Gaussian Distributions. The mixture weights represent the time proportions that those colors stay in the scene. The colors which can most probably be classified as background are those which stay for a longer time in the video segment and are more static.

MOG2:

It is also a "Gaussian Mixture-based Background/Foreground Segmentation Algorithm". One distinguishing and crucial attribute of this algorithm is that is chooses the number of Gaussian distribution most suitable for each pixel. It provides improved adaptability to changing scenes due to changes in illumination and other elements.

GMG:

This method combines both "statistical background estimation" and "per-pixel Bayesian segmentation". It uses the first few (default value-120) frames for background modeling. It uses "probabilistic foreground segmentation algorithm" that detects probable foreground objects using "Bayesian inference". Adaptive estimations are made; newer observations are given more weightage as compared to the older observations, in order to indulge the variable lighting. In order to remove noise, morphological operations are applied.

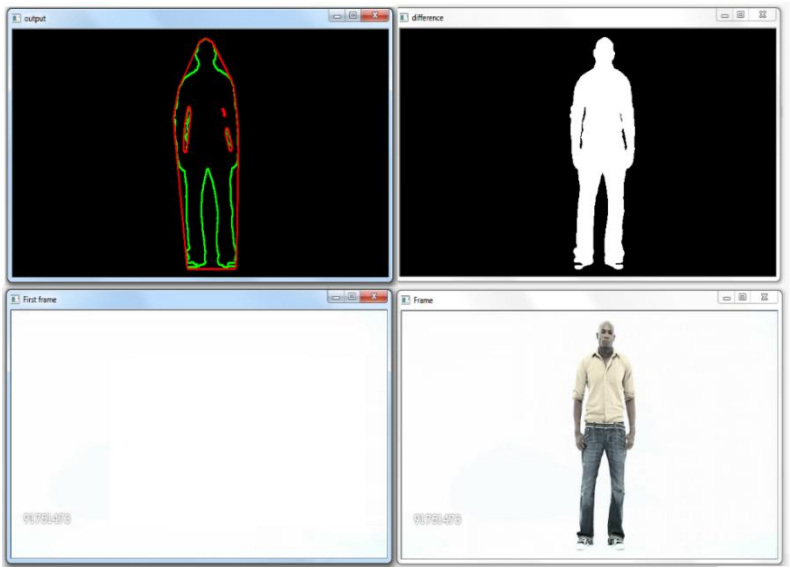


Fig 2.13 First frame Subtraction

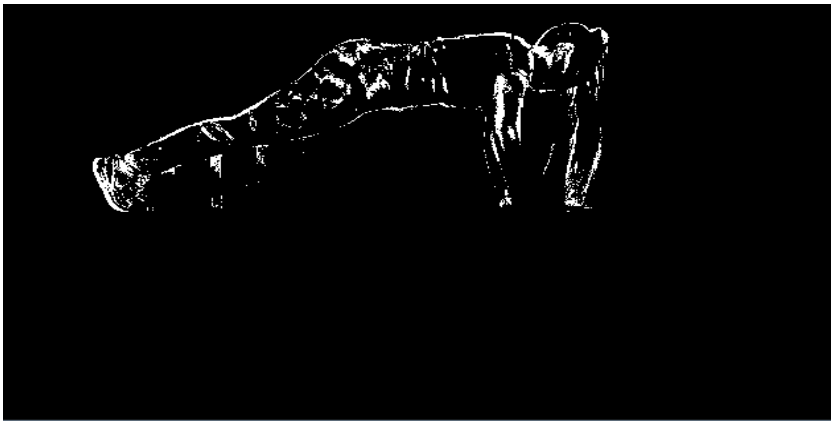


Fig 2.14 K-NN Background Subtraction

Fig 2.15 MOG2 Background Subtraction

2.3.4.3 GrabCut Algorithm

GrabCut algorithm [4] is an algorithm used for foreground extraction with limited user interaction. Initially, we start with a bounding box which is specified by the user as foreground. The algorithm approximates the color distribution of the required object and that of the background using a “Gaussian mixture model”. Using this, a “Markov random field” is constructed over the labels of the pixels, with a function for energy that desires areas that are connected and have the same label. Then a graph cut is run to infer their values.

The background elimination takes place as follows:

(1) The rectangle is input by the user. All the regions outside the user-defined rectangle is considered as definite background. Everything inside rectangle is unknown.

(2) The algorithm labels the foreground and background pixels in the initial stages depending on the data given by the user.

(3) A "Gaussian Mixture Model" (GMM) is used to model the foreground and background .

(4) Based on the user data, GMM learns and creates a new pixel distribution to label the undecided pixels as likely foreground or likely background in a way similar to clustering.

(5) Using this pixel distribution a graph is plotted with pixels as its nodes. Source node and Sink node are added such that the Source node connects every foreground pixel and the Sink node connects every background pixel.

(6) The probability of a pixel being foreground/background determines the edge weights of connecting pixels to source/end node. The weights between the pixels are given by the information about the edges or how similar the pixels are. Larger the difference in pixel color, lower will be the weight assigned to the edge between them.

(7) The graph is segmented using a mincut algorithm into Source node and Sink node graphs with minimum cost function. The cost function is "the sum of all weights of the edges that are cut". After segmentation, all the Source node connected pixels become foreground and the Sink node connected pixels become background.

(8) Classification convergence signifies the end of this iterative process.

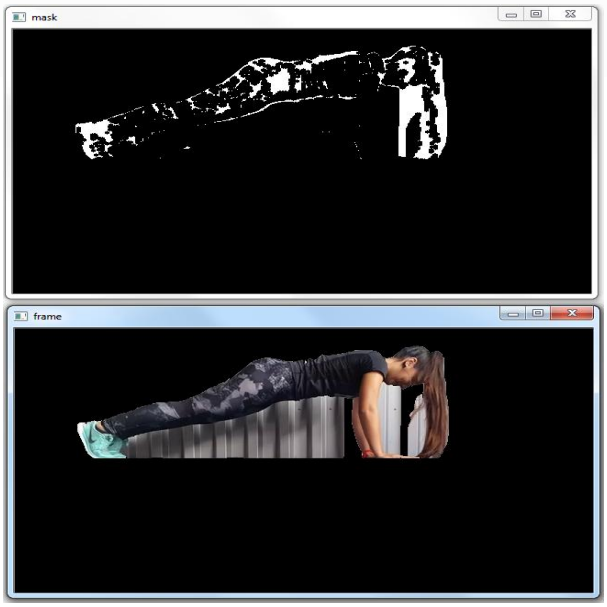


Fig 2.16 GrabCut algorithm

Algorithm	Advantages	Disadvantages
Canny Edge Detector	Provides good localization and better noise performance	Complex computations
Harris Corner Detector	Rotation Invariant	Partially variant to scale and affine intensity
Background Subtraction	Fast and Computationally light	Performance depends on quality of background used
HOG Gradient	Ease of implementation and Fast	Suitable for global features
Wavelet based segmentation	Better noise performance	Complex and computationally heavy

Table 2.1 Comparison of various feature detection algorithms

2.4 Conclusion

As mentioned in the above table, Canny Edge Detector Provides good localization and better noise performance but it is ineffective for complex computations. Algorithms such as background subtraction and HOG Gradient are fast to implement but while the former's performance depends on the background used, the latter is suited only for global features. Harris Corner Detector is rotation invariant but is susceptible partially to scale and affine intensity. Lastly, wavelet based segmentation is computationally heavy although offering a better performance under noise. Going forward, Canny Edge Detector is used for different applications.

Chapter 3

Skeletonization using Image Processing

3.1 Introduction

Skeletonization is a preprocessing operation which reduces the foreground in a binary image to binary objects that are 1 pixel wide thereby conserving the original region's connectivity while discarding most of the original foreground. It also provides region-based shape features. Thus, we obtain a skeleton of the original image [8].

3.2 Skeletonization

There are three major skeletonization techniques:

"Skeletonize": It performs consecutive iterations of the image. Each pass consists of identification of border pixels and removal based on the condition that the connectivity of the next adjacent object is not broken.

"Medial Axis Skeletonization": The object's medial axis has a set of all points with more than one point nearest to the object's boundary. Usually known as the topological skeleton, as its width is approximately 1-pixel but, containing the original object's connectivity.

"Morphological thinning": Removes pixels from the borders at each iteration without affecting the connectivity.

The two requirements to be fulfilled in order to render a "true skeleton" are:

(1) Topological

(2) Geometrical (to achieve translation, rotation, and scaling invariance while also maintaining the skeleton in the object's center.)

METHOD	Geometrical	Topological
Skeletonize	no	yes
Medial Axis	yes	yes
Thinning	no	yes

Table 3.1 Comparison of skeletonization techniques

(a) Medial Axis (b) Skeletonize (c) Thinning

Fig 3.1 Skeletonization

In addition to using the above approach, we implemented key point matching for the key points obtained using the feature descriptors.

3.3 Feature Descriptors

For proper detection, it is important to obtain features that are detectable especially the high-contrast regions of the image, such as object edges. Also, the relative positions between them in the original scene shouldn't be altered [6].

We use 3 main types of feature descriptors:

(1) SIFT

(2) SURF

(3) ORB

3.3.1 SIFT

SIFT is a feature detection algorithm prominently used for template matching and object recognition which plays a key role in computer vision [6]. The feature points obtained from a collection of images are stored in a database. Object Recognition in a new image compares each feature individually from the new image to the stored database and finds candidate matching features based on their feature vector's hamming distance. Subsets of key points that agree on the object and its location, scale, and orientation in the new image are a criteria based on which bad matches are filtered out. As a result of an efficient hash table implementation of the generalized Hough transform, consistent clusters are determined. Each cluster has 3 or more features that agree on the test image which undergoes further verification to discard any discontinuities. Based on accuracy of fit and amount of possible false matches, the probability that a particular set of features that indicate the presence of the object is obtained. Object matches that pass all these tests are correct with high confidence.

Algorithm:

"(1) Scale Space Extrema Detection: Detects points of interest, key points in terms of the SIFT model. Differences of successive Gaussian-blurred images are taken after convolution of the image with Gaussian filters at different scales. Key points are then assumed to be maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales.

(2) Key point Localization: Scale-space extrema detection produces a lot of key points, some being unstable. The next step performs a fit to the closest data for precise location, scale, and ratio of principal curvatures. This information allows low contrast points or badly localized points to be thrown away.

(3) Orientation Assignment: Each key point, based on local image gradient directions, is assigned one or more orientations. To become rotation invariant, the key point descriptor is represented relative to this orientation. For every pixel in a neighboring region around the key point in the Gaussian-blurred image L, the magnitude and direction calculations for the gradient are done. An orientation histogram with 36 bins is created each bin covering 10 degrees.

(4) Key point Descriptors: Key point locations at particular scales and assigned orientations to them were done previously. This ensured image location, scale and rotation invariance. Now, a distinctive and partially invariant descriptor vector has to be computed for each key point. This step is performed on the image closest to the key point's scale. "

Fig 3.2 Input Image

Fig 3.3 SIFT

3.3.2 SURF

SURF is a local feature detector and descriptor commonly be used for tasks such as object recognition, image registration, classification or 3D reconstruction and template matching [6]. SURF is several times faster than SIFT and robust against image transformations than the latter.

Integer approximation of determinant of Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image which is later used to detect interest points, is implemented in SURF. Its feature descriptor uses the sum of the Haar wavelet response around the point of interest. These can be computed using integral image also.

The image is transformed into coordinates using the multi-resolution pyramid technique, to copy the original image with Pyramidal Gaussian or Laplacian Pyramid shape to obtain an image with the same size and reduced bandwidth. This obtains a special blurring effect on the original image, called Scale-Space and ensuring scale invariance of the interest points.

Algorithm:

"(1) Detection: Square-shaped filters which act as a preprocessing stage of Gaussian smoothing. Filtering the image with a square is much faster if integral image is used. A blob detector based on the Hessian matrix is used to obtain points of interest. The determinant of the Hessian matrix is used to measure local change around the point and points where determinant has maximal value are taken into consideration.

(2) Descriptor: Providing a unique as well as robust description of an image feature, e.g., description of the intensity distribution of the pixels within the neighborhood of the point of interest is the main role of the descriptor. Most descriptors are computed in a local manner; hence a description is obtained for each point of interest identified. The descriptor's dimensionality has direct impact on its computational complexity as well as accuracy. A short descriptor could be robust against appearance variations, but will fail to define outliers clearly and thus give false positives."

Fig 3.4 SURF

3.3.3 ORB

ORB is a good alternative to SIFT and SURF in terms of computation cost and matching performance [6]. ORB is a combination of FAST key point detector and BRIEF descriptor with many modifications to enhance the performance. FAST finds key points, applies Harris corner measure to find top-most N points. It also produces multiscale-features because it uses pyramid.

The intensity weighted centroid of the patch with located corner at centre is also determined. The orientation of the key point is given by the direction of the vector from the corner point to the centroid. To improve the rotation invariance, moments are computed with x and y which should be in a circle of radius r, where r is the size of the patch.

The important property of BRIEF is that each bit feature has a large variance and a mean near 0.5. But after orientation along key point direction, this property is lost and becomes distributed. High variance makes a feature more discriminative, as the response to inputs is differential. Another desirable property is to have uncorrelated tests, since then each test will have a contribution to the result. A greedy search among all possible binary tests is run to find the ones that have high variance as well as means close to 0.5, as well as being uncorrelated to resolve all these.

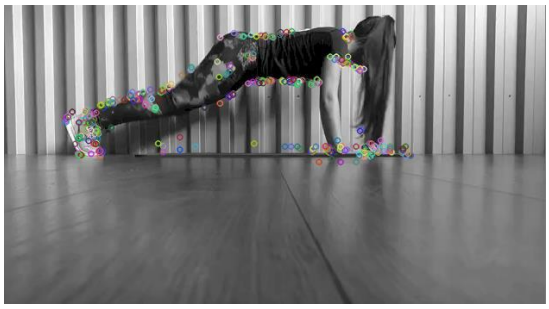


Fig 3.5 ORB with 500 features

Fig 3.6 ORB with 1500 features

Algorithm	Advantages	Disadvantages
SIFT	Invariance in orientation and scale	Complex and has a large computational overhead
SURF	More robust to scale changes	Requires more memory for descriptors
ORB	Comparatively faster and computationally light	Variance under orientation and scale

Table 3.2 Comparison of different feature descriptors

3.4 Key point Matching

There are two techniques used for key point matching:

(1) BF Matcher

(2) FLANN Matcher

3.4.1 BF Matcher

BF stands for Brute Force. It uses some distance calculation method to perform a descriptor match of features between different sets and the returns the closest match. We use ORB to obtain the features and hence the Hamming distance is used for calculation [4].

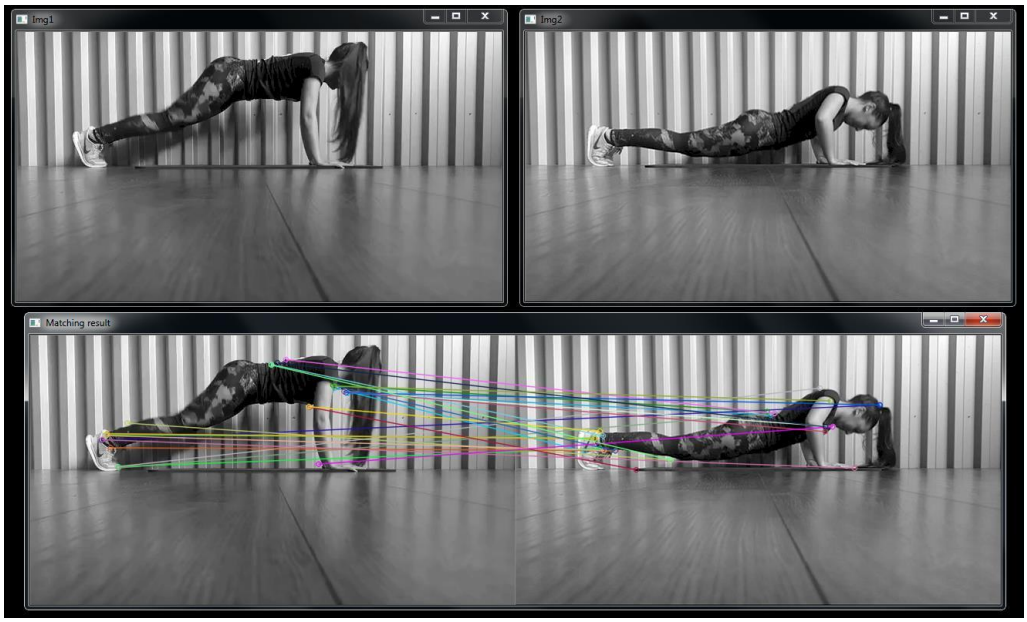


Fig 3.7 BF Matcher using ORB

3.4.2 FLANN Matcher

FLANN ("Fast Library for Approximate Nearest Neighbors") has a collection of algorithms optimized for "fast nearest neighbor search" in big datasets to extract high dimensional features. It is faster than BF Matcher for big datasets. For this matcher, we use SIFT to obtain features [4].

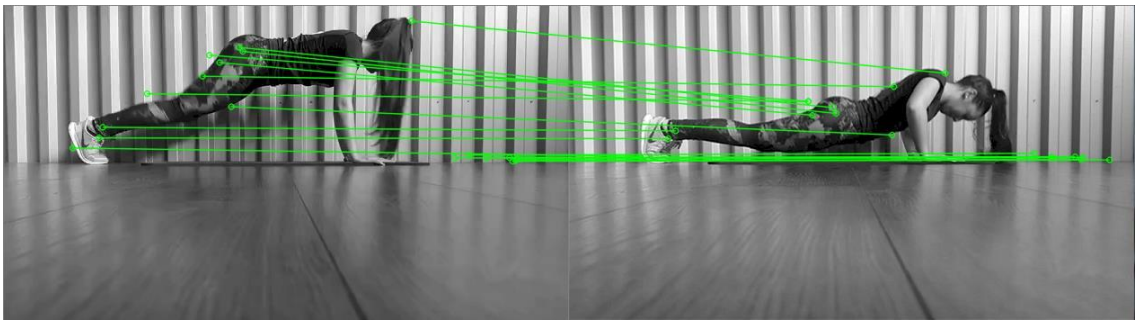


Fig 3.8 FLANN Matcher using SIFT

3.5 Conclusion

Skeletonization using image processing is an attempt to obtain the human vector build directly without any key points. It is effective if the input under test is an image although it is complex and computationally heavy.

The flow involved with using image processing involves getting the silhouette of the person, and then gradually thinning it down through many iterations. The downside of this approach being that to get the perfect skeleton, we need to have to the perfect silhouette which does not have a good repeatability rate.

The other approach used here is to try to obtain the human body's key points and try to match similar points as a means of comparison. The feature descriptors do a fairly good job in obtaining points over the human body but these points are not key points at necessary locations. The matching technique uses these feature descriptors to match similar points. This is a good template matching method but not good for frame comparisons which needed in our application. With different frames to compare with, the matching techniques fall well below the expectations and hence, are not the way to go forward.

Therefore, we need to fall upon a different approach which is explained in the next chapter.

Chapter 4

Skeletonization using Deep Learning Algorithm

4.1 Introduction

Human pose estimation largely deals with the predicament that is localizing anatomical key points (i.e.) “parts”, it is mainly focused on finding body parts of individuals. Inferring the pose of a person in an image especially socially engaged individuals, poses the following challenges:

(1) Each figure may contain a person who is located at any position or scale.

(2) Limb movements which make the part association cumbersome.

(3) Runtime complexity increases as the person count in the image increase, making real-time performance a tough task.

4.2 Pose Estimation

Pose estimation can be realized using two approaches namely: top-down approach and bottom-up approach. Let us now analyze the challenges that these approaches pose and why the latter approach is more robust compared to the former.

4.2.1 Top-Down Approach

According to [9], the top-down approach utilizes a detection module to identify humans and then apply the pose estimator for a single-person to locate the key points on the human body. The key benefit of top-down approaches is their competence in splitting a single complex task into multiple relatively easier tasks, i.e., “object detection and single-person pose estimation”. The object detector is adept in identifying small nominees which allows the pose estimator to excel in its performance with a focused regression space.

[9] uses models from deformable ConvNets which are pre-trained. In order to increase the recollection rate of the human nominees, experiments are conducted on validation sets of “PoseTrack 2017 and PoseTrack 2018” to decide which object detector is the best. Firstly, ground truth is inferred from the boxes bounding the human nominees from the interpreted key points, because in the dataset of PoseTrack 2017, the position of the bounding box is not given in the observations. Specifically, a bounding box from the minimum and maximum

coordinates of the 15 key points is located, and then the box is enlarged evenly by around 20%. Although the bounding boxes of ground truth are given in the dataset of PoseTrack 2018, a more reliable version based on the key points' ground truth locations is inferred. The training of the pose estimator is done using these inferred ground truth bounding boxes.

This approach directly leverages available methods for single-person pose estimation, but has to put up with early commitment, i.e., the person detector fails when people are in close proximity–there is no recuperation. Moreover, the runtime of these top-down approaches increases with more people detected leading to higher computational costs.

Let us now move on to the second approach (i.e.) bottom-up approach.

4.2.2 Bottom-Up Approach

The bottom-up approach detects human key points from all probable human nominees and then combines these key points into each individual's limbs based on numerous data association techniques. This is one of the main factors that make this approach more lucrative as they do not fall prey to early commitment and have the ability to reduce complexity of runtime due to various detected people.

For example, Pishchulin's pioneering work proposed a bottom-up approach which associated the jointly labeled part detection nominees to individual people. However, solving this problem is highly challenging and may take hours on an average for the processing to complete.

One such application of this approach is implemented by Cao, Simon, Wei and Sheikh which is known as OpenPose which is explained in detail.

4.3 OpenPose

4.3.1 Introduction

[10] presents an effective approach for "multi person pose estimation" with sophisticated precision on multiple public standards. It is a bottom-up depiction of "association scores via Part Affinity Fields (PAFs)", a set of 2D vector fields that encode the position and direction of limbs over the image domain. It is deduced that "these bottom-up representations of detection and association encode global context satisfactorily to allow a greedy parse to obtain high-quality results and with minimal computational cost".

4.3.2 Design

The system takes a color image of size l × b as input and produces the "2D locations of anatomical key points" for the detected human in the image as output. Firstly, a "feed-forward network" concurrently estimates a set of "2D confidence maps S" of the locations of different body parts and a set of "2D vector fields L of part affinities", which provides the extent of part linkages. The set S = (S_1, S_2 ... S_J) has J confidence maps, one per part, where $S_j \in R^{w \times h}$, $j \in \{1 \ldots J\}$. The set L = (L_1, L_2... L_C) has C vector fields, one per limb 1, where $L_c \in R^{w \times h \times 2}$, $c \in \{1 \ldots C\}$, each image location in L_c encodes a 2D vector (as shown in Fig 4.1). Lastly, the affinity fields and the confidence maps are parsed by greedy inference to output the 2D key points in Fig 4.1 .

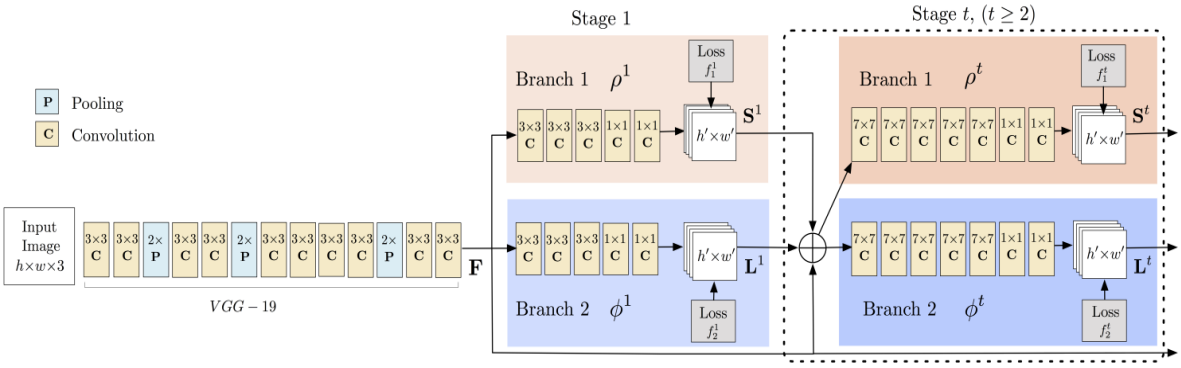


Fig 4.1 Network Architecture

4.3.3 Detection and Association

The network architecture (Fig 4.1), concurrently predicts affinity fields and detection confidence maps that encode association of parts. The network branches into two stages: the top branch (Branch 1) estimates the confidence maps whereas the bottom branch (Branch 2) estimates the affinity fields. Each branch is a repetitive prediction architecture, following Wei, which fine-tunes the predictions over successive stages, $t \in \{1. \ldots T\}$, with intermediate overseeing at each stage.

Initially the image is examined by a convolutional neural network which is set by the first 10 layers of VGG-19 and refined, creating a set of feature maps F which is fed as input to the first stage of each branch. In Stage 1, the network generates a set of detection confidence maps $S_1 = \rho_1$ (F) and a set of part affinity fields $L_1 = \varphi_1$ (F), where ρ_1 and φ_1 are the CNNs for deduction at the first stage . In each subsequent stage, the original image features F along

with the predictions from both the previous stage's branches are concatenated and utilized to obtain fine-tuned predictions.

$$\mathbf{S}^t = \rho^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \ \forall t \geq 2, \quad (4.1)$$

$$\mathbf{L}^t = \phi^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \ \forall t \geq 2, \quad (4.2)$$

, where ρ_t and φ_t are "the CNNs for Stage t's inference".

The next section explains how the detection and association in detail.

4.3.4 Method

VGG-19 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network has 19 layers and can classify images into 1000 object categories, such as food items, furniture, vehicles, and many animals. As a result, the network has learned rich feature representations for a wide range of images .

According to [11], "at the time of training, ConvNets' input is a fixed-size 224×224 RGB image. The only pre-processing done is subtracting each pixel by the mean RGB value computed on the training set. The image goes through a stack of convolutional layers, where filters with a very small receptive field 3×3 are used. One of the arrangements also utilizes 1×1 convolution filters, which is basically a linear transformation of the input channels. The convolution is fixed to 1 pixel; the spatial padding of convolutional layer input is done in a way which preserves the spatial resolution after convolution, i.e. 3×3 convolutional layers are padded by 1 pixel. Spatial pooling is done via five max-pooling layers, which follow few of the convolutional layers. Max-pooling is performed over a 2×2 pixel window, with stride 2".

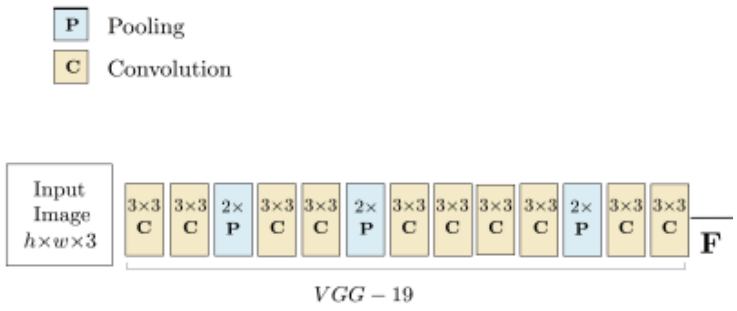


Fig 4.2 VGG-19

To supervise the network to iteratively estimate confidence maps of body parts in Branch 1 and PAFs in Branch 2, we apply two loss functions, one at each branch respectively at the end of each stage. We use an L_2 loss between the predictions and the ground truth fields and maps. At this stage, the loss functions are spatially weighted to deal with the problem of certain datasets not fully detecting all people. Specifically, the loss functions at both Stage t branches are:

$$f_{\mathbf{S}}^t = \sum_{j=1}^{J} \sum_{\mathbf{p}} \mathbf{W}(\mathbf{p}) \cdot \|\mathbf{S}_j^t(\mathbf{p}) - \mathbf{S}_j^*(\mathbf{p})\|_2^2, \tag{4.3}$$

$$f_{\mathbf{L}}^t = \sum_{c=1}^{C} \sum_{\mathbf{p}} \mathbf{W}(\mathbf{p}) \cdot \|\mathbf{L}_c^t(\mathbf{p}) - \mathbf{L}_c^*(\mathbf{p})\|_2^2, \tag{4.4}$$

, where S_j^* is "the ground truth part confidence map", L_c^* is "the ground truth part affinity vector field", W is "a binary mask" with W (p) = 0 when the interpretation is not found at an image location p. To avoid penalizing the true positive estimations during training, the mask is used. The intermediate guiding at each stage tackles the vanishing gradient problem by restoring the gradient from time to time. The general aim is:

$$f = \sum_{t=1}^{T} (f_{\mathbf{S}}^t + f_{\mathbf{L}}^t). \tag{4.5}$$

To evaluate f_S in Eq. (4.5) while training, we create the "ground truth confidence maps S*" from the interpreted 2D key points. Each confidence map is a 2D depiction of the conviction that a certain body part occurs at each pixel location. Ideally, if the image contains a single person, a single peak should exist in each confidence map if the analogous part is perceptible; if multiple people are detected, there has to be a peak corresponding to each perceptible part j for each person k. We first create discrete confidence maps $S_{j,k}^*$ for each person k. Let $x_{j,k} \in R^2$ be the ground truth position of body part j for person k in the image. The value at location $p \in R^2$ in $S_{j,k}$* is defined as,

$$\mathbf{S}_{j,k}^*(\mathbf{p}) = \exp\left(-\frac{||\mathbf{p} - \mathbf{x}_{j,k}||_2^2}{\sigma^2}\right), \tag{4.6}$$

Here, σ controls the extent of the peak. The ground truth confidence map to be estimated by the network is a collection of the discrete confidence maps via a max operator,

$$\mathbf{S}_j^*(\mathbf{p}) = \max_k \mathbf{S}_{j,k}^*(\mathbf{p}). \tag{4.7}$$

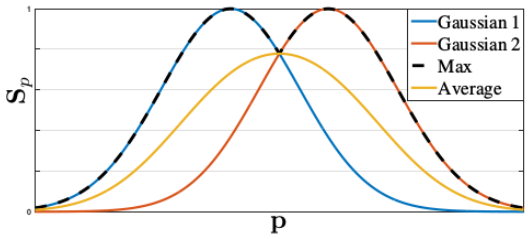


Fig 4.3 Significance of σ

During testing, the confidence maps are predicted and body part nominees are obtained by performing non-maximum suppression.

Fig 4.4 Confidence Map

Fig 4.5 Part Affinity Map

Given a set of detected body parts, how do we put together the full-body poses? We necessitate a confidence quantification that each pair of body part detections and their linkages belong to the same person. A likely measure of the linkages is to detect an extra point in-between limb part pairs, and check for its occurrence between candidate part detections. This results in two prominent challenges:

(1) When people crowd together, these midpoints are liable to false linkages.

(2) False linkages occur due to two limitations in the illustration (i.e.)

(2.1) only the location is encoded, and not the direction, of each limb.

(2.2) the limb's support region is reduced to one point.

To confront these drawbacks, a new feature representation called "part affinity fields" is presented that maintains both position and direction data across the limb's support region. The part affinity is a 2D vector field for each limb. For each pixel included in the region of a certain limb, a 2D vector encodes the orientation of the limb. Every limb has an analogous affinity field joining its two linked body parts.

Consider a single limb shown in Fig 4.6. Let $x_{j\,1,\,k}$ and $x_{j\,2,\,k}$ be "the ground truth positions of parts j_1 and j_2 from the limb c for the person k". If a point p lies on c, the value of $L_{c,\,k}^{*}(p)$ is "a unit vector that points from j_1 to j_2"; the p vector is zero-valued for all other points.

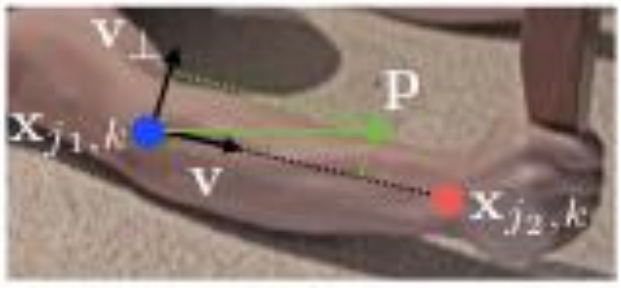


Fig 4.6 Part Association

To evaluate f_L in Eq 4.5 at the time of training, we define the "ground truth part affinity vector field", $L^{*}_{c,\,k}$, at an image point p as:

$$\mathbf{L}^{*}_{c,k}(\mathbf{p}) = \begin{cases} \mathbf{v} & \text{if } \mathbf{p} \text{ on limb } c, k \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{4.8}$$

Here, $v = (x_{j2,\,k} - x_{j1,\,k})/\|x_{j2,\,k} - x_{j1,\,k}\|_2$ is "the unit vector along the limb's orientation". The set of points on the limb is defined as those within a distance threshold of the line segment, i.e., those points p for which,

$$0 \le v \cdot (p - x_{\,j1,\,k}) \le l_{c,\,k} \text{ and } |v\perp \cdot (p - x_{j1,\,k})| \le \sigma_{l,} \tag{4.9}$$

, where the width of the limb σ_l is a distance in pixels and the length of the limb is given by $l_{c,k} = \|x_{j2\,,k} - x_{j1\,,k}\|_2$, and $v_\perp$ is a vector perpendicular to v.The ground truth part affinity field gives the average of all the people's affinity fields in the image,

$$\mathbf{L}^{*}_{c}(\mathbf{p}) = \frac{1}{n_c(\mathbf{p})} \sum_{k} \mathbf{L}^{*}_{c,k}(\mathbf{p}), \tag{4.10}$$

In Eq 4.10, n_c (p) is "the number of non-zero vectors at point p across all k people (i.e., the average at pixels where limbs of different people overlap)". During testing, we calculate the ordering of the estimated PAF with the nominated limb formed by linking the body parts that are detected.

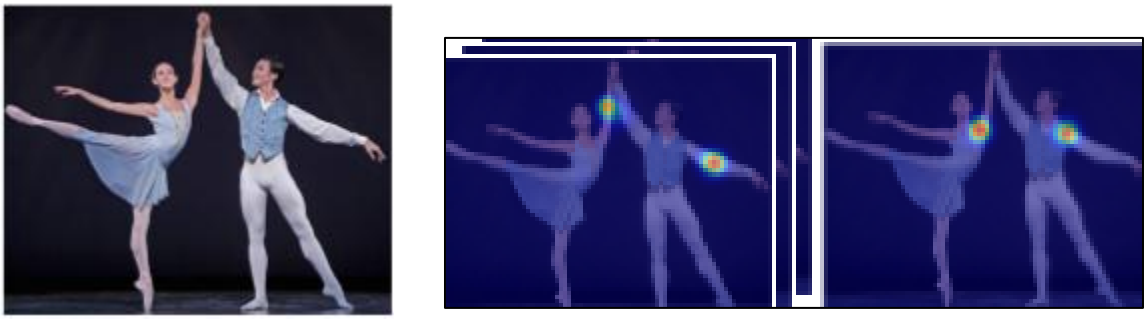
Specifically, for two candidate part locations d_{j1} and d_{j2}, we sample the PAF, L_c along the line segment to quantify the linkages' confidence.

$$E = \int_{u=0}^{u=1} \mathbf{L}_c\left(\mathbf{p}(u)\right) \cdot \frac{\mathbf{d}_{j_2} - \mathbf{d}_{j_1}}{||\mathbf{d}_{j_2} - \mathbf{d}_{j_1}||_2} du, \tag{4.11}$$

Here, p (u) interpolates the location of the two body parts d_{j1} and $d_{j2,}$

$$p\ (u) = (1 - u)\ dj1 + ud_{j2} \tag{4.12}$$

In practice, the integral is estimated by sampling and summing uniformly-spaced values of u.



(a) Input Image (b) Part Confidence Maps

(c) Part Affinity Fields (d) Output Image

Fig 4.7 Human Vector Build

4.4 Datasets

The implementation of the OpenPose algorithm was done on two datasets namely:

(1) COCO

(2) MPII

Let's have a look at these datasets briefly:

4.4.1 COCO

" 0 - Nose, 1 - Neck, 2 - Right Shoulder, 3 - Right Elbow, 4 - Right Wrist, 5 - Left Shoulder, 6 - Left Elbow, 7 - Left Wrist, 8 - Right Hip, 9 - Right Knee, 10 - Right Ankle, 11 - Left Hip, 12 - Left Knee, 13 - Left Ankle, 14 - Right Eye, 15 - Left Eye, 16 - Right Ear, 17 - Left Ear, 18 – Background "

4.4.2 MPII

" 0 - Head, 1 - Neck, 2 - Right Shoulder, 3 - Right Elbow, 4 - Right Wrist, 5 - Left Shoulder, 6 - Left Elbow, 7 - Left Wrist, 8 - Right Hip, 9 - Right Knee, 10 - Right Ankle, 11 - Left Hip, 12 - Left Knee, 13 - Left Ankle, 14 - Chest, 15 - Background "

The primary difference between the two is that MPI has 15 points while COCO has 18 points.

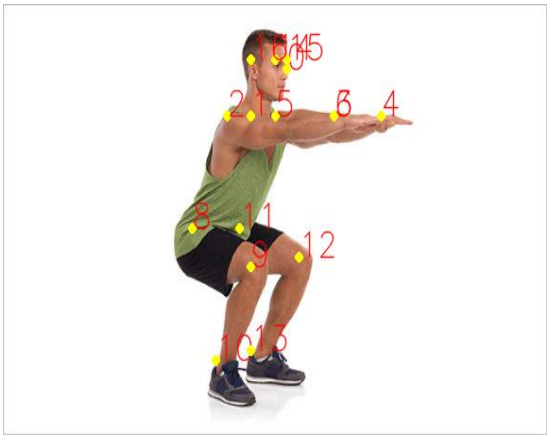


Fig 4.8 COCO dataset key points

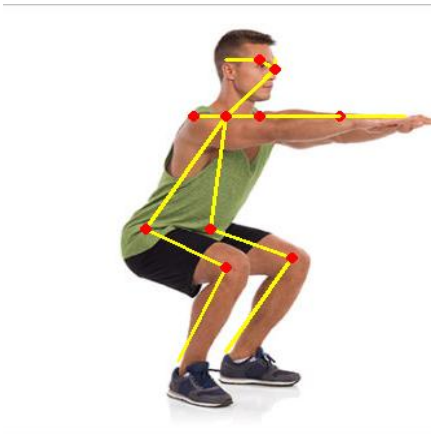


Fig 4.9 COCO dataset vector build

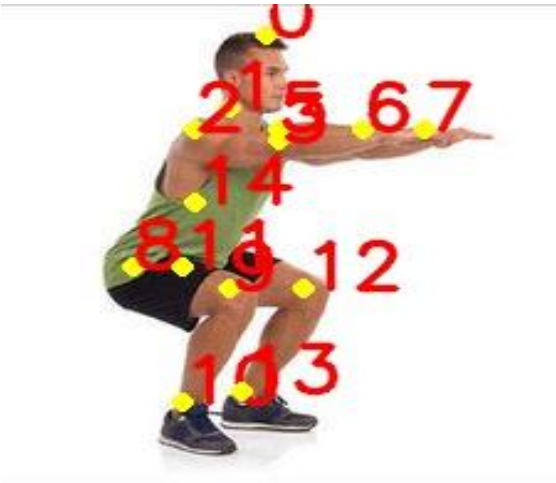


Fig 4.10 MPII dataset key points

Fig 4.11 MPII dataset vector build

4.5 Conclusion

OpenPose algorithm makes minimal adjustments with the input, the noise behaviour is more robust and the implementation is sparse allowing the computational power to come down. Since this algorithm is able to identify the person, his/her body parts and associate it with the same person; the performance under noise is avant-garde. The efficiency of this algorithm is high since it follows the bottom-up approach, thereby localizing the key points precisely and allowing for a more accurate comparison with reference.

Coming to the datasets, while the key points and the vector build for both COCO and MPII datasets are very good, for our application, we need certain points positioned at essential joints on the human body. MPII provides a point on the back which helps a great deal in analyzing the posture more precisely. Notice that point 14 in the MPII dataset is the back point which is of utmost importance to us. This gives the MPII dataset an edge over COCO dataset which is primarily used in facial recognition applications (due to the presence of 5 key points on the face) .Therefore; we will be making use of the MPII dataset for our application.

Chapter 5

Comparison with Reference and Correction

5.1 Reference Comparison

5.1.1 Introduction

For our application, the objective is to compare the given test case with an ideal case and provide necessary correction. The main focus while comparing the test with the reference model is to check the angles between the joints of interest. From the previous chapter, we obtained the various key points on the human body (silhouette) representing the essential joints. The key points were joined in the appropriate manner to give a visualization of the human skeleton via the human vector build. The commodities obtained in the previous chapter can be utilized for comparison. For this purpose, two methods are most convenient: Hough Transform and Co-ordinate method.

Let us see the implementation of these methods.

5.1.2 Hough Transform

Hough Transform is a general method to identify any shape, if that shape can be represented in mathematical form. The shape can be detected even if it is broken or slightly distorted. We use it for a line for our application.

The first approach used for comparison is the Hough transform. For this, we utilize the human vector build as a whole to detect angles. A recurring issue in computer image processing is straight line detection in digitized images. Primarily, the image has many discrete points lying on a plain background. Detecting the various points is not much of a predicament but to identify the occurrence of groups of collinear or nearly collinear points, therein lies the problem. Rather than finding collinear points, Hough proposed a different method. According to [12], this technique obtains a straight line in a parameter space by converting each of the figure points.

The parameter space is defined by the parametric representation used to describe lines in the image. A histogram array is constructed representing the parameter space. For each parameter pair of r and theta, we find, in the input image, the number of non-zero pixels that are in the vicinity of the corresponding line, and aptly increment the array at position (r,

theta). Each non-zero pixel is essentially "voting" for probable line nominees. The local maxima in the ensuing histogram give us an indication of the parameters of the lines which are most probable.

A binary edge map is taken as input by the Hough transform to find edges located as straight lines. The concept of the Hough transform is that every edge point in the edge map is changed to all probable lines that could pass through that point. A line in Hough space is formed by each edge point and the regions where most Hough lines intersect is interpreted as true lines in the edge map.

The standard Hough transform identifies lines with parameters r and theta and no information about its length. As we get infinite lines, the Hough transform is a little too slow. To obtain lines of finite length, supplementary exploration needs to be done to decide which regions of the image contribute to each line. The Hough Transform can be made faster and can be used to find finite lines by "Progressive Probabilistic Hough Transform" [13]. It is based on the hypothesis that "using a random subset of voting points give a good estimate to the actual result, and those lines can be obtained during the voting process by following the path of connected components". This returns the two ends of each line segment.

A Canny Edge detector is initially used to detect all the edges in the image. The Probabilistic Hough transform is the applied on this result to obtain a final output of detected lines from the original image.

The main problem with this algorithm is that the results from several runs may differ. This is more likely if many lines share the same pixels. This approach also detects the lines based on the usage of edge detection. This leads to detection of the boundaries of the vector build and not the vector build itself no matter how thin the lines are.

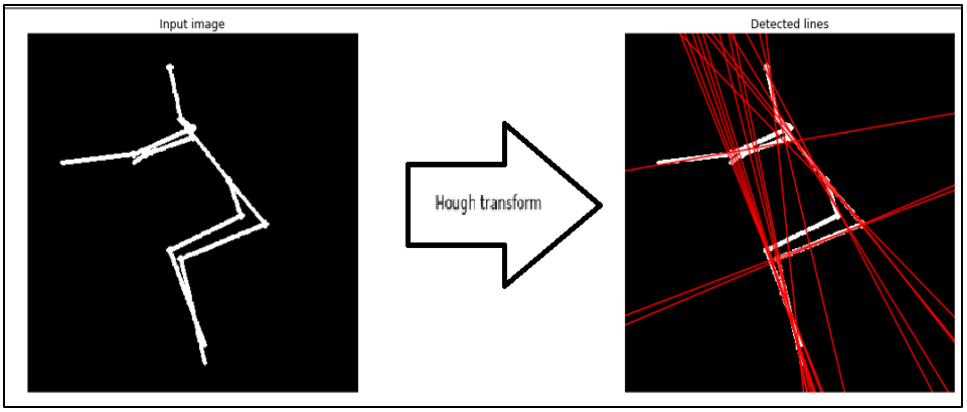


Fig 5.1 Hough Transform

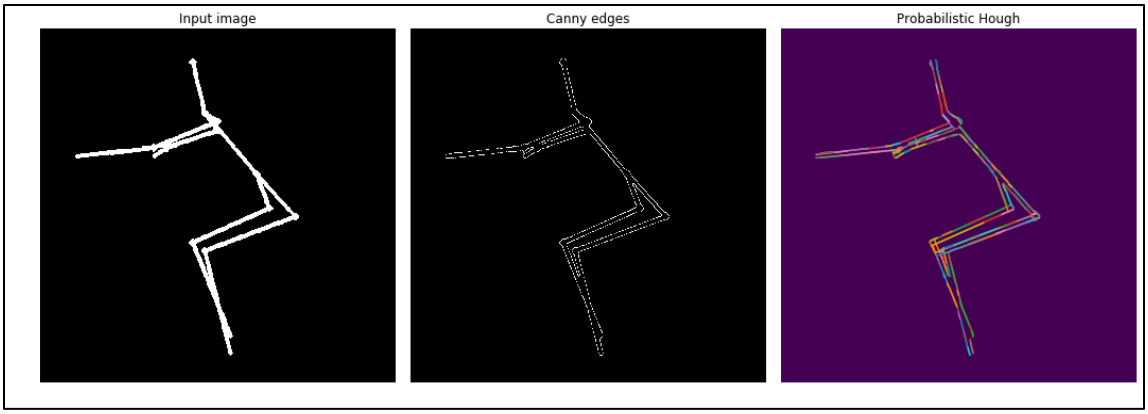


Fig 5.2 Probabilistic Hough Lines

5.1.3 Co-ordinate Method:

The classic method of using the co-ordinate axes values to determine the angles of various line segments on the human vector.

The second approach for comparison is done using the key points themselves. The co-ordinate method is the direct method taken out of mathematics textbooks. Given the x-axis and y-axis values of the two ends of a required line segment, the angle of that line segment can be obtained with respect to the x-axis. In our application, the co-ordinates of various joints are obtained by the OpenPose [10] function. This information can be used to find the angles of various body parts such as the back, hip and knee. The angles between two line segments can also be found by relative subtraction.

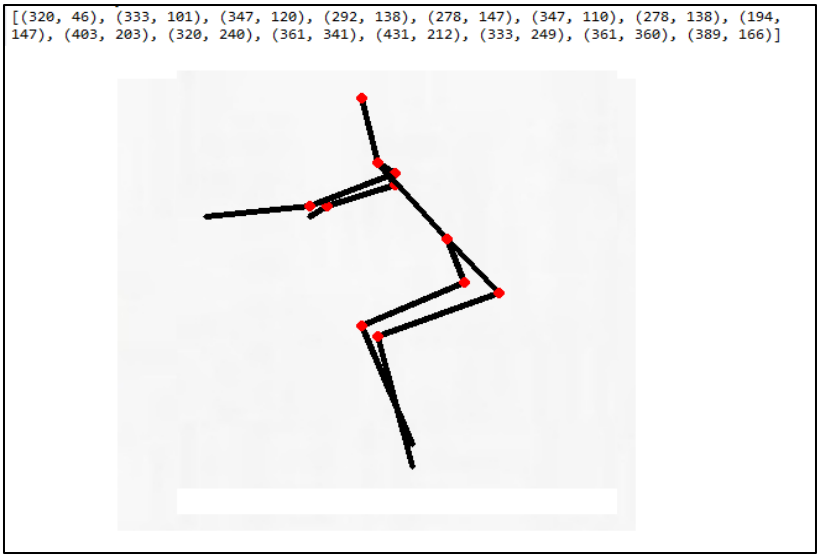


Fig 5.3 Co-ordinates of various joints obtained via MPII dataset

After comparing with reference, correction is a necessary step forward in case the posture is wrong.

This important step is emphasized next.

5.2 Comparison and Correction

After obtaining the angles between various joints, the utilization of angles is the next foray in our application. For all exercises, the main motive is to get the body fit and healthy. In order to make full use of these exercises, the posture has to be performed in the way it is meant to be done. This implies the various angles in which the body joints are bent. The most obvious point of interest is the back. In most postures, the back has to be straight for it to be deemed good. The other key angles which need to be considered for posture correction are the hip angles and the knee angles.

While the knee angle is utilized for specific postures, the hip angle, while it has a prerequisite similar to knee angle, also has an additional task of determining the lowest points of any given posture.

There are many postures which exist and each posture calls for a different set of angle rules. No matter what the pose, the analysis of angles can be performed provided the view of the human vector build is a side view. This allows for the proper usage of all available joint angles and hence paves way for an accurate comparison. For our application, we have delved into five different postures which are quite common; squats, dead lifts, planks, bridges and lunges. Each of these poses have different demands to be deemed ideal and our application works around this ideality. As mentioned earlier, regardless of the posture, the one constant is a straight back and that is true for the aforementioned five poses as well. The hip being straight is a necessity for two of these poses; planks and bridges, whereas the plank also calls for a straight knee. The squat and dead lift depend upon how much a person crouches whilst maintaining a straight back. While the squat takes both, the hip and knee angles into consideration, the main motive of dead lift is to keep the glute as far as possible from the head and hence the hip angle is the main factor. For a perfect lunge, a few more conditions have to be met. Along with a straight back, the back has to be perpendicular with respect to the ground. The fore shin also should maintain perpendicularity with the ground.

All these different conditions are a necessity for the postures to be ideal. A user can self train oneself but the caveat being the poses being wrong which could prove to be dangerous and could potentially affect the body frame during the long run. The minor arches of back or hip can cause problems to the body if it isn't corrected. This calls for a trainer and our application provides the user a personal trainer without another individual's physical presence. The user can run the video of the required squat and the necessary frames are extracted. These frames have to be the lowest point of the exercise which is where the ideality is compared. The angles obtained from the select frames and then compared with an ideal reference. The user then gets the feedback of whether the posture is a good or a bad one along with necessary angles. For a bad posture, the correction is provided so as to make the pose as ideal as possible.

The corrections to be made are shown in 2 phases:

(1) Frame-wise correction data

(2) Overall Correction data

When a person is performing a particular exercise, some aspects of his posture such as whether or not he is maintaining a straight back at every stage, or for a video in every frame can be checked immediately and does not require a range of motion to be checked. And other aspects such as at least the hip is being bent to a particular threshold, can only be checked after the person has finished his range of motion, only after this can we detect and if needed correct the persons posture across the complete motion.

With reference to the 5 different exercises i.e., Squat, Dead lift, Lunge, Glute Bridge and Plank, the frame wise correction data is whether or not the person maintains a perfectly straight back.

The overall correction varies from exercise to exercise:

(1) Squat: If the person's hip angle as well as the knee angle, moves within the range of angles obtained from our reference. Also checks if the person's back is straight at every frame.

(2) Dead lift: If the person torso, moves within the range of angles obtained from our reference. Also checks if the person's back is straight at every frame.

(3) Lunge: If the person's knees move within the range of angles obtained from our reference. Also checks if the persons back is straight at every frame.

(4) Glute Bridge: If the person's hip moves within the range of angles obtained from our reference. Also checks if the persons back is straight at every frame.

(5) Planks: Checks if the person's back, hip and knee is straight.

The output-snippet below, shows how the frame-wise correction is provided typically,

```
frame  7
[19.500793850483575, 88.69061018058945, 148.94572888364723, 19.500793850483575, 105.33819614050536,
174.8394275891264]
Keep a straight back
Just 19.500793850483575  degrees for a perfectly straight back
frame  8
[5.701308454464225, 75.16124937459642, 149.21585347367355, 19.500793850483575, 102.98913236774324,
177.23987591961628]
frame  9
[20.480357311809257, 59.21585347367355, 149.21585347367355, 4.534961410886353, 89.18964697172389,
171.71539570381583]
frame  10
[10.911128384283415, 73.41266144307258, 163.41266144307258, 10.911128384283415, 77.37416312109269,
176.62027586665593]
Keep a straight back
Just 10.911128384283415  degrees for a perfectly straight back
frame  11
[10.911128384283415, 59.21585347367355, 136.59001659476624, 10.911128384283415, 75.59331484356312,
174.06445947657795]
Keep a straight back
Just 10.911128384283415  degrees for a perfectly straight back
```

Fig 5.4 Example of Frame-wise correction data

As the output shows, in the 5 frames checked, in 2 frames namely 7 and 11, the back is not completely straight, and provides the angle by which the correction is to be made to get a perfectly straight back.

The overall correction data gives an insight as to where the person is erroneous, and how it can be corrected, by pointing out, which area in the body, the posture is going wrong.

5.2.1 Squat

The figures shown below (Fig 5.5, Fig 5.6) represent the output for an ideal video of a person performing 3 squats in the video. One of these squats has been deliberately performed incorrectly in order to check the code's effectiveness. We plot the person's hip angle as a function of frames, to see how it varies as the video progresses. The ideal squats are characterized by the person's hip angle varying in a periodic nature.

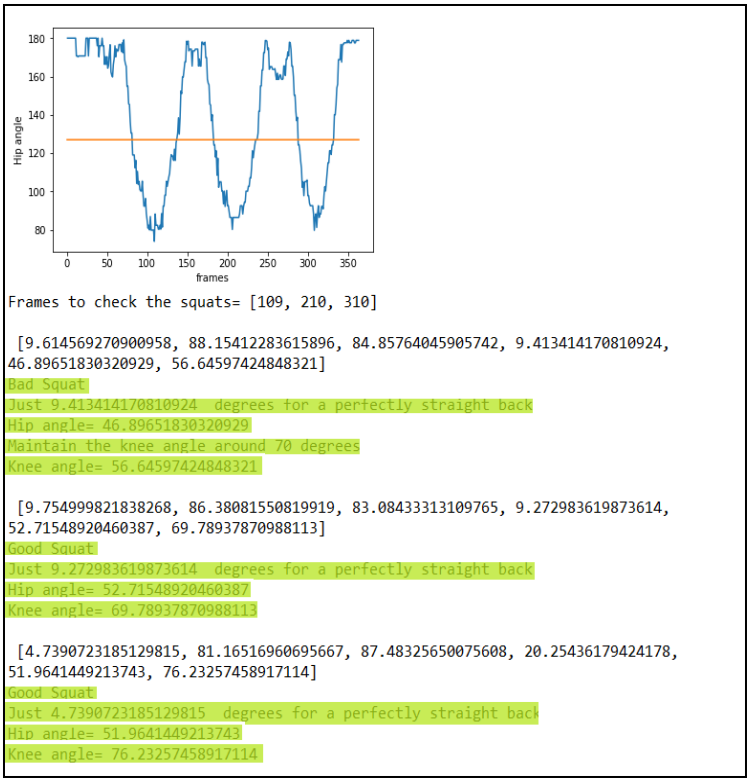


Fig 5.5 Output for a video with 3 squats performed

(a) Frame 109 (b) Frame 210 (c) Frame 310

Fig 5.6 Lowest points of the squats in the video

As seen in Fig 5.5, the code identifies that 3 squats have been performed and 1 of the 3 is detected as a bad squat, as expected, and it provides the correction needed to get the required posture. The remaining 2 squats have been checked as well, and as we can see, they have been characterized a good squat, as the angles meet the required values.

5.2.2 Dead lift

The figures shown below (Fig 5.7, Fig 5.8) represent the output for an ideal video of a person performing 4 dead lifts in the video. One of these dead lifts has been deliberately performed incorrectly in order to check the code's effectiveness. We plot the person's hip angle as a function of frames, to see how it varies as the video progresses. The ideal dead lifts are characterized by the person's hip angle varying in a periodic nature.

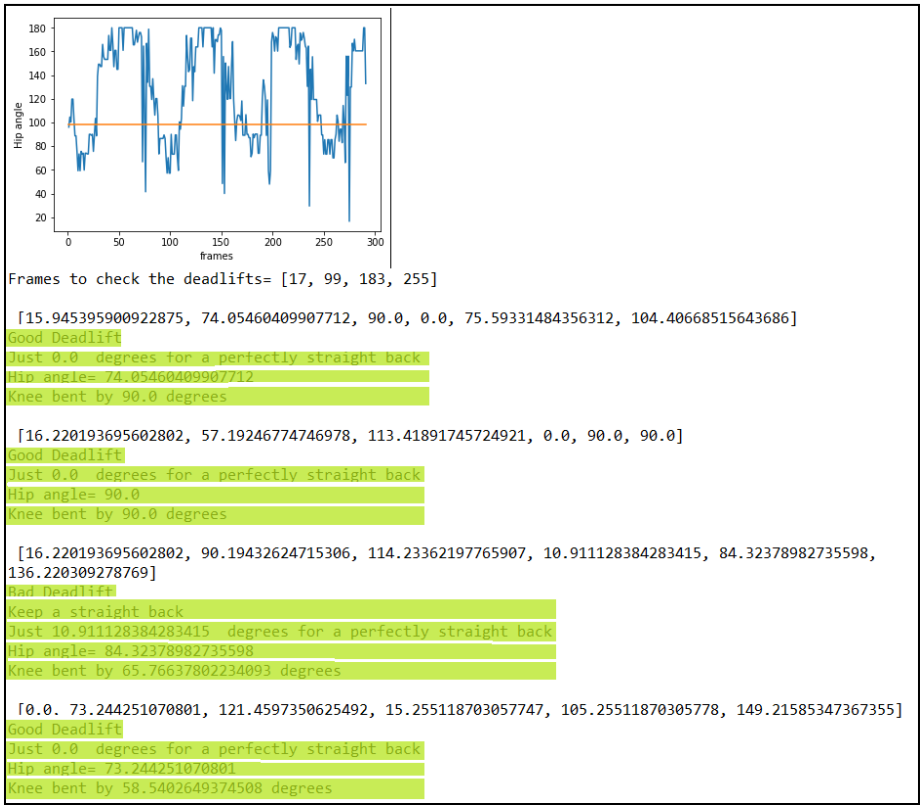
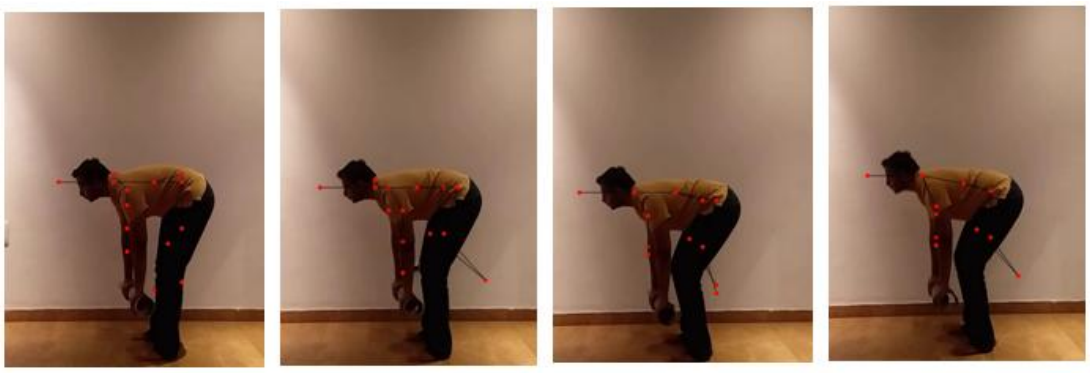


Fig 5.7 Output for a video with 4 dead lifts performed



(a) Frame 17 (b) Frame 99 (c) Frame 183 (d) Frame 255

Fig 5.8 Lowest points of the dead lifts in the video

As seen in Fig 5.7, the code identifies that 4 dead lifts have been performed and 1 of the 4 is detected as a bad dead lift, as expected, and it provides the correction needed to get the required posture. The remaining 3 dead lifts have been checked as well, and as we can see, they have been characterized a good dead lift, as the angles meet the required values.

5.2.3 Lunge

The figures shown below (Fig 5.9, Fig 5.10) represent the output for a video of a person performing 4 lunges in the video. We plot the person's hind shin angle as a function of frames, to see how it varies as the video progresses. The ideal lunges are characterized by the person's hind shin angle varying in a periodic nature.

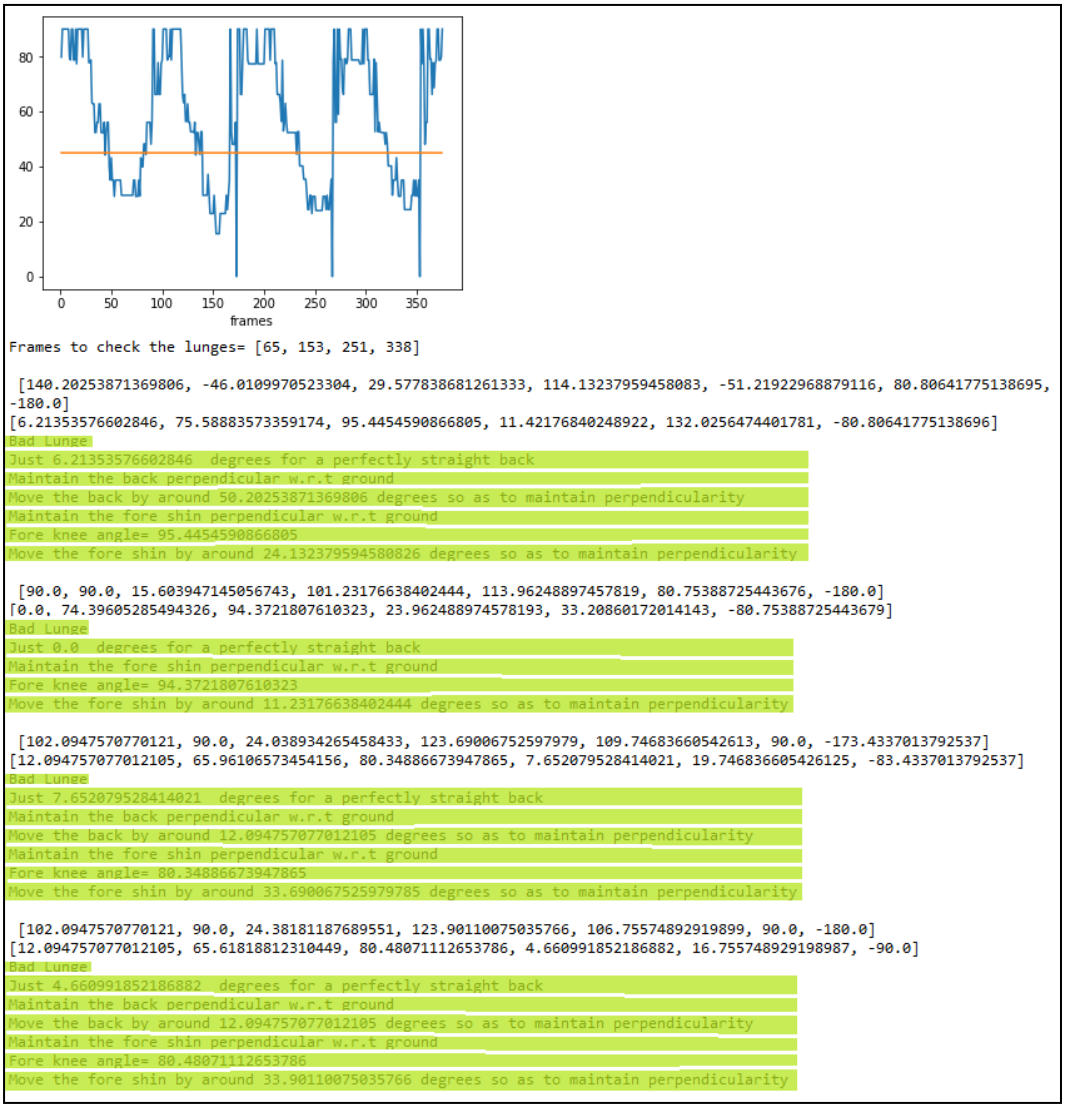


Fig 5.9 Output for a video with 4 lunges performed

(a) Frame 65 (b) Frame 153 (c) Frame 251 (d) Frame 338

Fig 5.10 Lowest points of the lunges in the video

As seen in Fig 5.9, the code identifies that 4 lunges have been performed. In this video, all the 4 lunges were performed with some imperfection and this can be seen from the results obtained. All the 4 lunges are detected as bad lunges as expected, and it provides the correction needed to get the required posture.

5.2.4 Glute Bridge

The figures shown below (Fig 5.11, Fig 5.12) represent the output for an ideal video of a person performing 4 glute bridges in the video. One of these bridges has been deliberately performed incorrectly in order to check the code's effectiveness. We plot the person's hip angle as a function of frames, to see how it varies as the video progresses. The ideal bridges are characterized by the person's hip angle varying in a periodic nature.

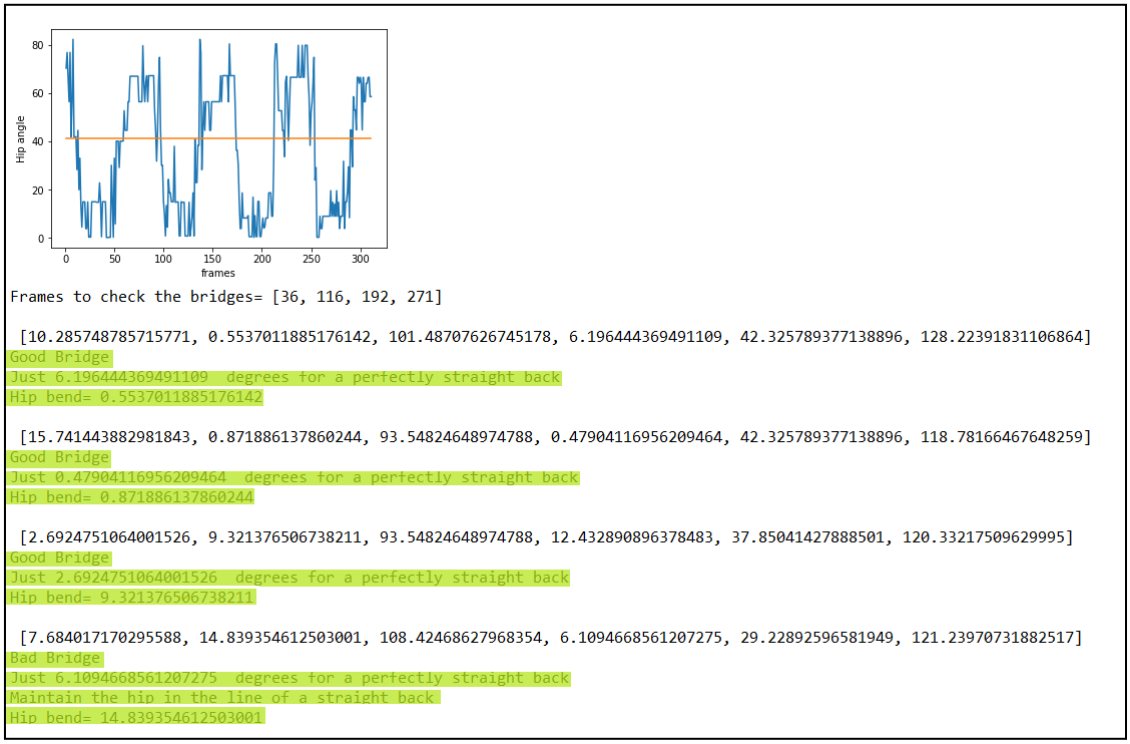


Fig 5.11 Output for a video with 4 glute bridges performed

(a) Frame 36 (b) Frame 116 (c) Frame 192 (d) Frame 271

Fig 5.12 Highest points of the glute bridges in the video

As seen in Fig 5.11, the code identifies that 4 bridges have been performed and 1 of the 4 is detected as a bad bridge, as expected, and it provides the correction needed to get the required posture. The remaining 3 glute bridges have been checked as well, and as we can see, they have been characterized a good bridge, as the angles meet the required values.

5.2.5 Plank

The figures shown below (Fig 5.13, Fig 5.14) represent the output for an ideal video of a person performing planks. Plank is an exercise which needs to be tracked in every frame as the criterion is a straight body (back, hip and knee). So, the whole video runs by with a frame by frame comparison of the video with the ideal case. The snippet shown below contains 3 frames of the planking video which show the various aspects of planks.

```
[3.0268221250656495, 7.648626840285829, 13.818471663155094, 21.679658619478573, 15.855061761668082,
4.553144497838105]
Good Plank
Just 3.0268221250656495  degrees for a perfectly straight back
Knee bend= 4.553144497838105
Hip bend= 7.648626840285829
frame  233
[3.0268221250656495, 7.648626840285829, 13.818471663155094, 21.679658619478573, 15.855061761668082,
5.656314480967495]
Good Plank
Just 3.0268221250656495  degrees for a perfectly straight back
Knee bend= 5.656314480967495
Hip bend= 7.648626840285829
frame  234
[3.0268221250656495, 7.648626840285829, 13.818471663155094, 21.679658619478573, 15.855061761668082,
4.553144497838105]
Good Plank
Just 3.0268221250656495  degrees for a perfectly straight back
Knee bend= 4.553144497838105
Hip bend= 7.648626840285829
frame  235
```

Fig 5.13 Output for a plank video (3 frames)

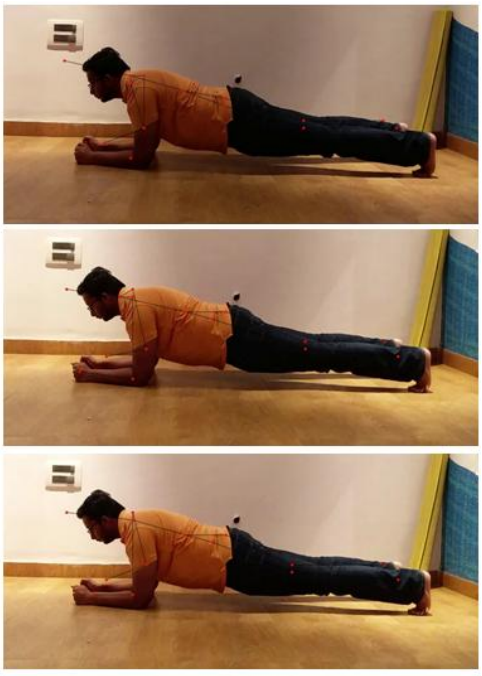


Fig 5.14 Consecutive frames of 3 planks in the video

As seen in Fig 5.13, of the 3 chosen frames, all 3 have been characterized as good planks as the angles meet the required criteria. The main criterion in this posture is that all the 3 essential angles have to be straight and in a bad case, the required corrections will be provided based on ideality.

5.3 Conclusion

The main idea behind comparison and correction is to make sure the person gets the desired results without him/her getting hurt. For this purpose, it is important to have one key parameter which can be put under the scanner, which in our case, is the joint angles. As seen in the images above, the knee and the hip angles are of utmost importance as they are the key points of analysis for most of the exercises. But, the most crucial aspect is maintaining a straight back which is very significant for the body weight to be distributed evenly. Correction is provided so that the person knows where he/she is erring, and can correct himself/herself in the future.

Chapter 6

Future Work and Conclusions

6.1 Future Work

The scope for this platform is immense with applications in the medical field for rehabilitations and in the field of sports and fitness.

Since, the computation cost for this algorithm is high, optimization of this platform can be looked into not only for implementing it for real time applications but for implementing it on a mobile platform as well.

To increase the usage of the algorithm, work can be done to make this algorithm universal and not restricted to identification of a set of poses.

6.2 Conclusion

While the initial idea was to achieve the objective using image processing, as the project progressed it was seen that the computational cost kept going up, and the results showed no further improvement. This gives a fair idea about the computational power required to implement the same technique on a video file.

As an alternative, we implemented feature descriptors to attain key points, further apply key point matching to get the desired output. Here as well, the computational cost was a constraint and it also required that the exercises should be performed and be captured at the same rate. More importantly, the key points obtained were placed randomly on the human body which was a hindrance for our application which requires accurate results.

Thus, a literature review was undertaken to find out other ways to reach the goal. While reviewing the various papers on this subject, we came across the OpenPose algorithm.

The OpenPose algorithm based on deep-neural network was sparse and robust and required minimal adjustments for implementing on our platform. It made use of certain aspects of the image processing approach and utilized this to obtain the human vector with the highest accuracy possible.

While the OpenPose algorithm provided us with the key points in the person's body, it did not provide us any means to determine if the person's posture was improper or not. Hence, in order to achieve this, we added extra functionality to provide us with the angles made by different parts of the body such as the angle between the torso and thigh, the angle between the thigh and the shin.

With the additional functionality of angle measurements, we were able to track the body movement of the person, and were able to determine what parameters would constitute a good posture. Once these parameters were obtained, we were able to determine if the persons posture was proper or not and provide corrections necessary to get the proper posture.

REFERENCES

[1] G. Papadopoulos, A. Axenopoulos and P. Daras, "Real-Time Skeleton-Tracking-Based Human Action Recognition Using Kinect Data," in *Springer-Verlag New York, Inc.*, New York, 2014.

[2] W. Ao-yu, T. Min and D. Jin-xiang, "A survey of silhouette detection techniques for non-photorealistic rendering," in *Third International Conference on Image and Graphics (ICIG'04)*, 2004.

[3] U. Jaliy, H. Parekh and D. Thakore, "A Survey on Object Detection and Tracking Methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970-2978, 2014.

[4] A. Mordvintsev and K.Abid, "OpenCV-Python Tutorials," OpenCV, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io.

[5] "Wikipedia," Wikimedia Foundation, Inc., 2001. [Online]. Available: https://en.wikipedia.org.

[6] M. Hassaballah, A. Ali and H. Alshazly, Image Features Detection, Description and Matching, 2016.

[7] R. Zhang and J. Ding, "Object Tracking and Detecting Based on Adaptive Background Subtraction," *Procedia Engineering*, vol. 29, pp. 1351-1355, 2012.

[8] J. K.Lakshmi and M.Punithavalli, "A Survey on Skeletons in Digital Image Processing," in *2009 International Conference on Digital Image Processing*, 2009.

[9] G. Ning, P. Liu, X. Fan and C. Zhang, "A Top-down Approach to Articulated Human Pose Estimation and Tracking," *CoRR*, vol. abs/1901.07680, 2019.

[10] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *CoRR*, vol. abs/1611.08050, 2016.

[11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CVPR*, vol. arXiv 1409.1556, 2014.

[12] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, vol. 15, no. 0001-0782, pp. 11-15, 1972.

[13] F. Masci, "Line Detection by Hough transformation," 20 April 2009. [Online]. Available: http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf. [Accessed 24 July 2012].

thesis

ORIGINALITY REPORT

12%	9%	8%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	2%
2	en.wikipedia.org Internet Source	1%
3	export.arxiv.org Internet Source	1%
4	"Computer Vision – ECCV 2018 Workshops", Springer Nature, 2019 Publication	1%
5	www.scribd.com Internet Source	1%
6	ijarcce.com Internet Source	<1%
7	Submitted to Coventry University Student Paper	<1%
8	Submitted to University of Southampton Student Paper	<1%
9	Submitted to Sheffield Hallam University	