"YOGA POSTURE CORRECTION USING MACHINE LEARNING UNDER VARYING POSES"

MINI PROJECT REPORT

Submitted by

Kshitija Kalbande(6CE30)
Shraddha kamble(6CE53)
Harshika Lende(6CE23)
Isha Kawale(6CE24)

to

Rashtrasant Tukadoji Maharaj Nagpur University

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

Bachelor of Engineering in Computer Engineering

DEPARTMENT OF COMPUTER ENGINEERING

CUMMINS COLLEGE OF ENGINEERING FOR WOMEN
NAGPUR-441110 2021-2022

CERTIFICATE

This is to certify that the project report entitled **"Yoga Posture Correction Using Machine Learning Under varying poses"** is submitted by "**Kshitija kalbande"** in partial fulfillment of the requirements for the award of the Degree of *Bachelor of ENGINEERING in COMPUTER ENGINEERING.* to the RASHTRASANT TUKADOJI MAHARAJ NAGPUR University as a record of work done by her under our supervision and guidance.

Guide:
Prof. Pinky Gangwani
Department of CE

Head of the Department:
Prof. Sharayu Deote
Department of CE

Internal Examiner

External Examiner

ACKNOWLEDGMENT

We express our sincere gratitude to all those people who have been associated with this project and have held us with it and made it a worth while experience. We extend our thanks to various people who have shared their opinion and experiences through which we received the required information crucial for our project.

We would like to express our special thanks of gratitude to our principle"**Dr. Bharat Bhushan Joshi**" as well as our HOD "**prof. Sharayu Deote**" who gave us the golden opportunity to this wonderful project on the topic "Yoga Posture Correction Using Machine Learning under Varying Poses" which also helped us in doing a lot of research and we came to know about so many new things. We are really thankful to them.

We would like to extend our sincere thanks to our project guide "**Prof. Pinky Gangwani**" who gave us this opportunity to learn the subject in a practical approach and gave us valuable suggestions regarding the project.

ABSTRACT

Yoga's popularity is growing on a daily basis since the covid-19 has had such a significant impact on everybody’s lives. The basis for this is the numerous physical, mental, and spiritual benefits that yoga may provide. Many people are following this trend and practise yoga without the help of a professional. However, doing yoga incorrectly or without sufficient direction can be harmful to one's health. Till date very limited literature is present on this challenging issue.

The issues are that the presented approaches demonstrate low accuracy in similar positions, the existing datasets consist of limited yoga poses, and the user has to be in the specified range of the camera . To overcome the mentioned issues, in this paper, we are proposing a yoga posture correction using a deep learning model to support users and customers with the proper guidance for yoga. The main objective of the proposed work is to include more positions and try to give better accuracy for similar poses, as previous models only have limited positions and the accuracy for the similar position is low.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE
1.	**INTRODUCTION**	
	1.1 Project outline	5
	1.2 Data Flow Diagram	6
	1.3 Technologies	7
	1.3.1 C++	8
	1.3.2 python	8
	1.3.3 parameter	8
	1.3.4 image point	8
	1.3.5 Camera matrix	8
	1.3.6 Flags	9
	1.4 Codes	11
	1.4.1 PYTHON	11
	1.4.2 C++	14
	1.5 Output	17
	1.6 Working	19
	1.7Conclusion	24
2.	**LITERATURE REVIEW**	**25**
	2.1 Estimation Of Yoga Posture	25
	2.2 Reference	26

1. INTRODUCTION

The application provides the yoga posture detection and correction intended use of the application is to correct improper postures of the yoga. The application will aim to guide the user to do yoga in a safe way and achieve the best results from yoga. Some people can do yoga regularly and they really feel very happy in daily life. There are various ways through which one can learn yoga,yoga can be learnt by attending the classes at a yoga center or through home tutoring it can also be self-learned with the help of some electronic devices like, computer, video and books. Most people prefer self learning but it is hard for them to find the incorrect parts of their yoga poses and themselves.Every year many people are affected from various types of musculoskeletal disorders due to accidents or aging problems yoga can promote positive physical change. An application of pose estimation which has attracted many researchers in this field is exercise and fitness. One form of exercise with intricate postures is yoga which is an age-old exercise that started in India but is now famous worldwide because of its many spiritual, physical and mental benefits. The problem with yoga however is that, just like any other exercise, it is of utmost importance to practice it correctly as any incorrect posture during a yoga session can be unproductive and possibly detrimental. This leads to the necessity of having an instructor to supervise the session and correct the individual's posture. Since not all users have access or resources to an instructor, an artificial intelligence-based application might be used to identify yoga poses and provide personalized feedback to help individuals improve their form.

1.1 PROJECT OUTLINE

As the covid-19 has made a huge impact on our life, the popularity of yoga is increasing daily.The reason for this is the physical, mental benefits that could be obtained by practicing yoga. Many are following this trend and practicing yoga without the training of an expert .However, practicing yoga in an improper way will lead to bad health issues such as strokes, nerve damage etc. So, practicing proper yoga postures is an important factor that is to be considered. We will include more positions and try to give accuracy for similar poses because previous models only have limited positions and the accuracy for the similar position is low.
The proposed Yoga Posture Correction System mainly consists of two components. They are,

- Keypoints Detection using OpenCV
- Higher Probability Prediction & Comparison

The overall workflow of the system is as follows. The user's movements are captured and streamed in real-time to the system using a media streaming server. Then the system detects the joints or the keypoints of the user with the use of a pose estimation, either OpenCV.After keypoints are detected by the pose estimation methods, those are sent to the yoga pose detection module. Using the keypoints data obtained from the previous components it was possible to predict the user's yoga pose before the user reaches the finalphase of the asana,using the movements that's required to be done to reach that stage, as the current model is able to detect keypoints only.

1.2 Data Flow Diagram

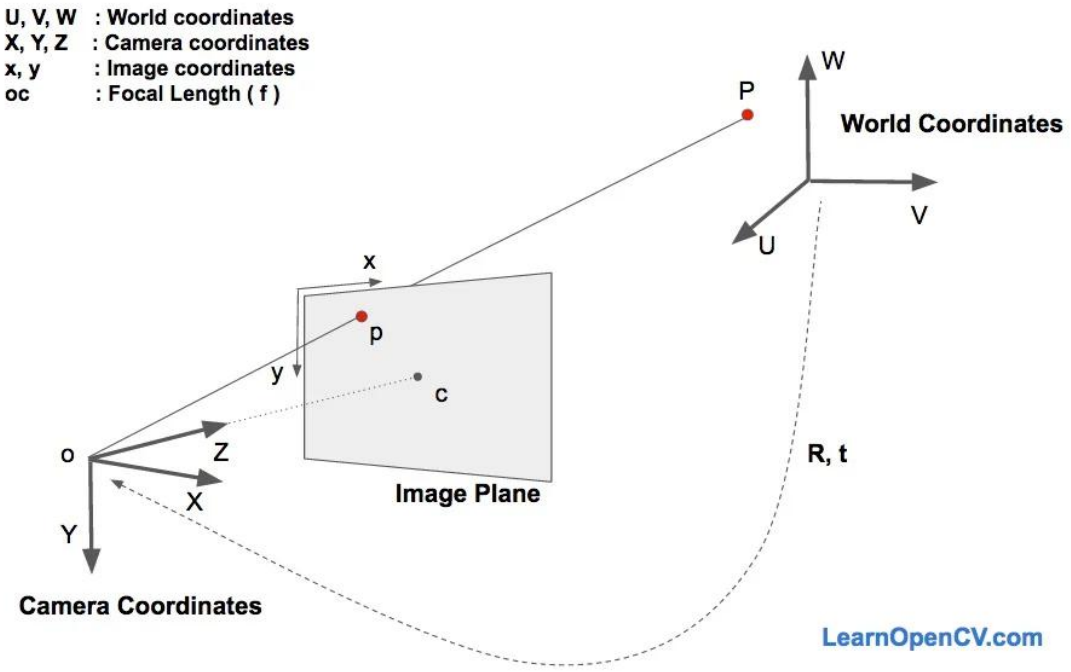


Fig 1.1 Pose Estimation Diagram

1.3 Technologies

Key Point Detection Using Open CV :

OpenCV Library in python, which stands for Open Source Computer Vision, is a trendy library used for achieving artificial intelligence through python. Using the OpenCV library, we can process real-time images and videos for recognition and detection.
When humans see a particular human or animal's image, we can recognize that image even when its orientation is changed or rotated. The reason behind it is that, as humans, we can easily identify the crucial keypoints in a given image. Thus, even if the image is rotated, we can still identify and relate it with the original picture because our brain can figure out the key points. Similarly, we need key points while processing images on the computer to identify the important points from an image. We achieve that by using OpenCV keypoint. While a computer is performing image processing, the computer should be able to identify similar characteristics in a given image irrespective of the transformations and rotations it goes through.
The computer should even be able to find similarities between different images belonging to a similar category. This is possible by observing the key points in a given image.

C++: bool solvePnP(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess=false, int flags=SOLVEPNP_ITERATIVE)

Python: cv2.solvePnP(objectPoints, imagePoints, cameraMatrix, distCoeffs[, rvec[, tvec[, useExtrinsicGuess[, flags]]]]) → retval, rvec, tvec.

Parameters: objectPoints – Array of object points in the world coordinate space. I usually pass vector of N 3D points. You can also pass Mat of size Nx3 (or 3xN) single channel matrix, or Nx1 (or 1xN) 3 channel matrix. I would highly recommend using a vector instead.

imagePoints – Array of corresponding image points. You should pass a vector of N 2D points. But you may also pass 2xN (or Nx2) 1-channel or 1xN (or Nx1) 2-channel Mat, where N is the number of points.

cameraMatrix – Input camera matrix $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$. Note that f_x, f_y can be approximated by the image width in pixels under certain circumstances, and the c_x and c_y can distCoeffs – Input vector of distortion coefficients (k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6],[s_1, s_2, s_3, s_4]]) of 4, 5, 8 or 12 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed. Unless you are working with a Go-Pro like camera where the distortion is huge, we can simply set this to NULL. If you are working with a lens with high distortion, I recommend doing a full camera calibration.

rvec – Output rotation vector.

tvec – Output translation vector.

useExtrinsicGuess – Parameter used for SOLVEPNP_ITERATIVE. If true (1), the function uses the provided rvec and tvec values as initial approximations of the rotation and translation vectors, respectively, and further optimizes them.

flags – Method for solving a PnP problem:

SOLVEPNP_ITERATIVE Iterative method is based on Levenberg-Marquardt optimization. In this case, the function finds such a pose that minimizes reprojection error, that is the sum of squared distances between the observed projections imagePoints and the projected (using projectPoints()) objectPoints .

SOLVEPNP_P3P Method is based on the paper of X.S. Gao, X.-R. Hou, J. Tang, H.-F. Chang “Complete Solution Classification for the Perspective-Three-Point Problem”. In this case, the function requires exactly four object and image points. SOLVEPNP_EPNP Method has been introduced by F.Moreno-Noguer, V.Lepetit and P.Fua in the paper “EPnP:

Efficient Perspective-n-Point Camera Pose Estimation". The flags below are only available for OpenCV 3.

SOLVEPNP_DLS Method is based on the paper of Joel A. Hesch and Stergios I. Roumeliotis. "A Direct Least-Squares (DLS) Method for PnP".SOLVEPNP_UPNP Method is based on the paper of A.Penate-Sanchez, J.Andrade-Cetto, F.Moreno-Noguer. "Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation". In this case the function also estimates the parameters f_x and f_y assuming that both have the same value. Then the cameraMatrix is updated with the estimated focal length.

1.4 CODES

PYTHON:

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
from skimage import io
# im = cv2.imread("headPose.png");
# size = im.shape
im = cv2.imread("headPose (1).webp");
size = im.shape
image_points = np.array([
                            (359, 391),     # Nose tip
                            (399, 561),     # Chin
                            (337, 297),     # Left eye left
corner
                            (513, 301),     # Right eye
right corne
                            (345, 465),     # Left Mouth
corner
                            (453, 469)      # Right mouth
corner
                        ], dtype="double")
model_points = np.array([
                            (0.0, 0.0, 0.0),              #
Nose tip
                            (0.0, -330.0, -65.0),         #
Chin
                            (-225.0, 170.0, -135.0),      #
Left eye left corner
                            (225.0, 170.0, -135.0),       #
Right eye right corne
                            (-150.0, -150.0, -125.0),     #
Left Mouth corner
                            (150.0, -150.0, -125.0)       #
Right mouth corner
                        ])
```

```
focal_length = size[1]

center = (size[1]/2, size[0]/2)

camera_matrix = np.array(
                         [[focal_length, 0, center[0]],
                         [0, focal_length, center[1]],
                         [0, 0, 1]], dtype = "double"
                         )
print("Camera Matrix :\n {0}".format(camera_matrix))
NameError                                 Traceback (most
recent call last)
<ipython-input-48-7ba112858ad0> in <module>()
----> 1 print("Camera Matrix :\n {0}".format(camera_matrix))

NameError: name 'camera_matrix' is not defined
#This is used for stack overflow
ist_coeffs = np.zeros((4,1)) # Assuming no lens distortion

(success, rotation_vector, translation_vector) =
cv2.solvePnP(model_points, image_points, camera_matrix,
dist_coeffs, flags=cv2.cv2.SOLVEPNP_ITERATIVE)
print("Rotation Vector:\n {0}".format(rotation_vector))
NameError                                 Traceback (most
recent call last)
<ipython-input-50-e95e755557c1> in <module>()
----> 1 print("Rotation Vector:\n
{0}".format(rotation_vector))

NameError: name 'rotation_vector' is not defined
print("Translation Vector:\n {0}".format(translation_vector))
NameError                                 Traceback (most
recent call last)
<ipython-input-51-2e06d2d9391e> in <module>()
----> 1 print("Translation Vector:\n
{0}".format(translation_vector))

NameError: name 'translation_vector' is not defined
nose_end_point2D, jacobian) =
cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)]),
```

```
rotation_vector, translation_vector, camera_matrix, dfor p
in image_points:
     cv2.circle(im, (int(p[0]), int(p[1])), 3, (0,0,255), -1)

p1 = ( int(image_points[0][0]), int(image_points[0][1]))
p2 = ( int(nose_end_point2D[0][0][0]),
int(nose_end_point2D[0][0][1]))
cv2.line(im, p1, p2, (255,0,0), 2)
# Display image
cv2_imshow(im)
cv2.waitKey(0)
```

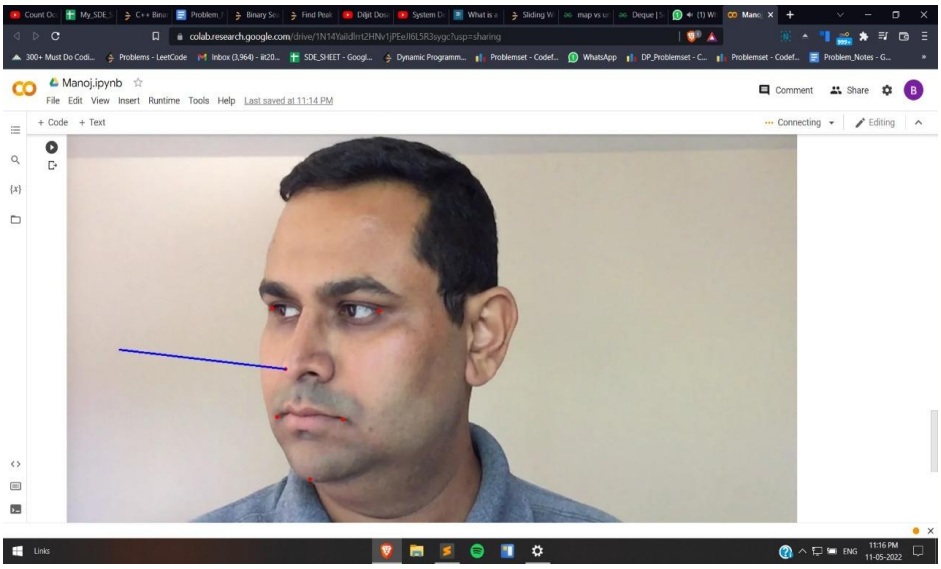
C++ :

```
#include <opencv2/opencv.hpp>

using namespace std;
using namespace cv;

int main(int argc, char **argv)
{

    // Read input image
    cv::Mat im = cv::imread("headPose.jpg");

    // 2D image points. If you change the image, you need to
change vector
    std::vector<cv::Point2d> image_points;
    image_points.push_back( cv::Point2d(359, 391) );    //
Nose tip
    image_points.push_back( cv::Point2d(399, 561) );    //
Chin
    image_points.push_back( cv::Point2d(337, 297) );     //
Left eye left corner
    image_points.push_back( cv::Point2d(513, 301) );    //
Right eye right corner
    image_points.push_back( cv::Point2d(345, 465) );    //
Left Mouth corner
    image_points.push_back( cv::Point2d(453, 469) );    //
Right mouth corner

    // 3D model points.
    std::vector<cv::Point3d> model_points;
    model_points.push_back(cv::Point3d(0.0f, 0.0f, 0.0f));
// Nose tip
    model_points.push_back(cv::Point3d(0.0f, -330.0f, -
65.0f));          // Chin
    model_points.push_back(cv::Point3d(-225.0f, 170.0f, -
135.0f));       // Left eye left corner
    model_points.push_back(cv::Point3d(225.0f, 170.0f, -
135.0f));        // Right eye right corner
```

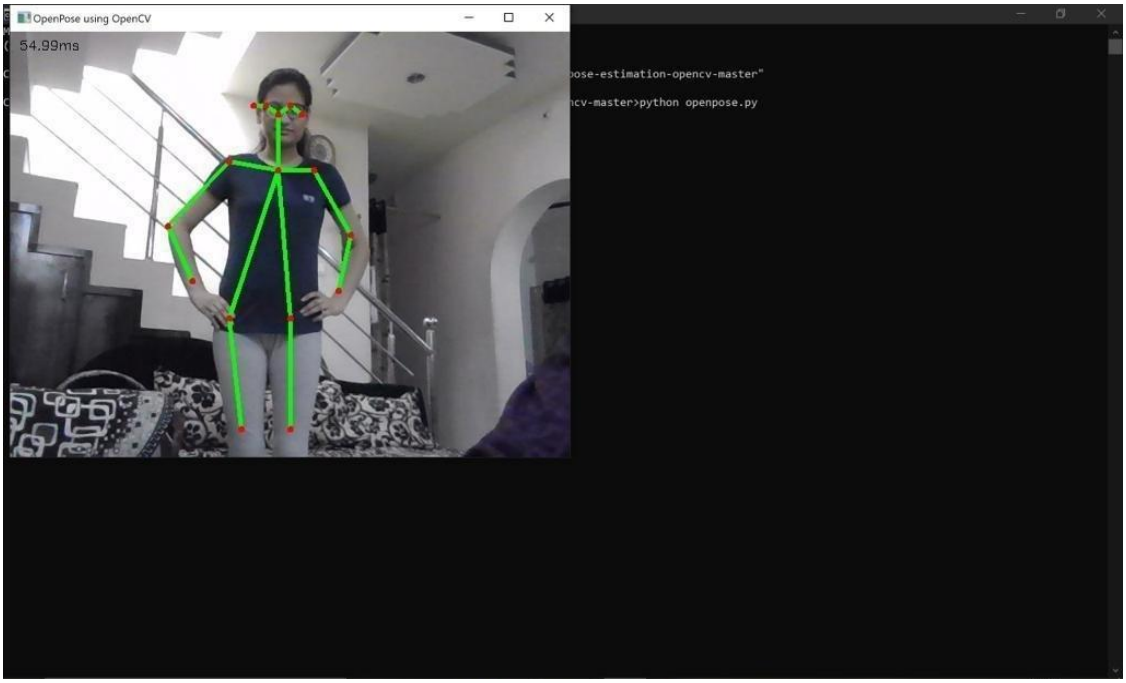
```
    model_points.push_back(cv::Point3d(-150.0f, -150.0f, -
125.0f));      // Left Mouth corner
    model_points.push_back(cv::Point3d(150.0f, -150.0f, -
125.0f));       // Right mouth corner

    // Camera internals
    double focal_length = im.cols; // Approximate focal
length.
    Point2d center = cv::Point2d(im.cols/2,im.rows/2);
    cv::Mat camera_matrix = (cv::Mat_<double>(3,3) <<
focal_length, 0, center.x, 0 , focal_length, center.y, 0, 0,
1);
    cv::Mat dist_coeffs =
cv::Mat::zeros(4,1,cv::DataType<double>::type); // Assuming
no lens distortion

    cout << "Camera Matrix " << endl << camera_matrix <<
endl ;
    // Output rotation and translation
    cv::Mat rotation_vector; // Rotation in axis-angle form
    cv::Mat translation_vector;

    // Solve for pose
    cv::solvePnP(model_points, image_points, camera_matrix,
dist_coeffs, rotation_vector, translation_vector);

    // Project a 3D point (0, 0, 1000.0) onto the image
plane.
    // We use this to draw a line sticking out of the nose

    vector<Point3d> nose_end_point3D;
    vector<Point2d> nose_end_point2D;
    nose_end_point3D.push_back(Point3d(0,0,1000.0));

    projectPoints(nose_end_point3D, rotation_vector,
translation_vector, camera_matrix, dist_coeffs,
nose_end_point2D);

    for(int i=0; i < image_points.size(); i++)
    {
```

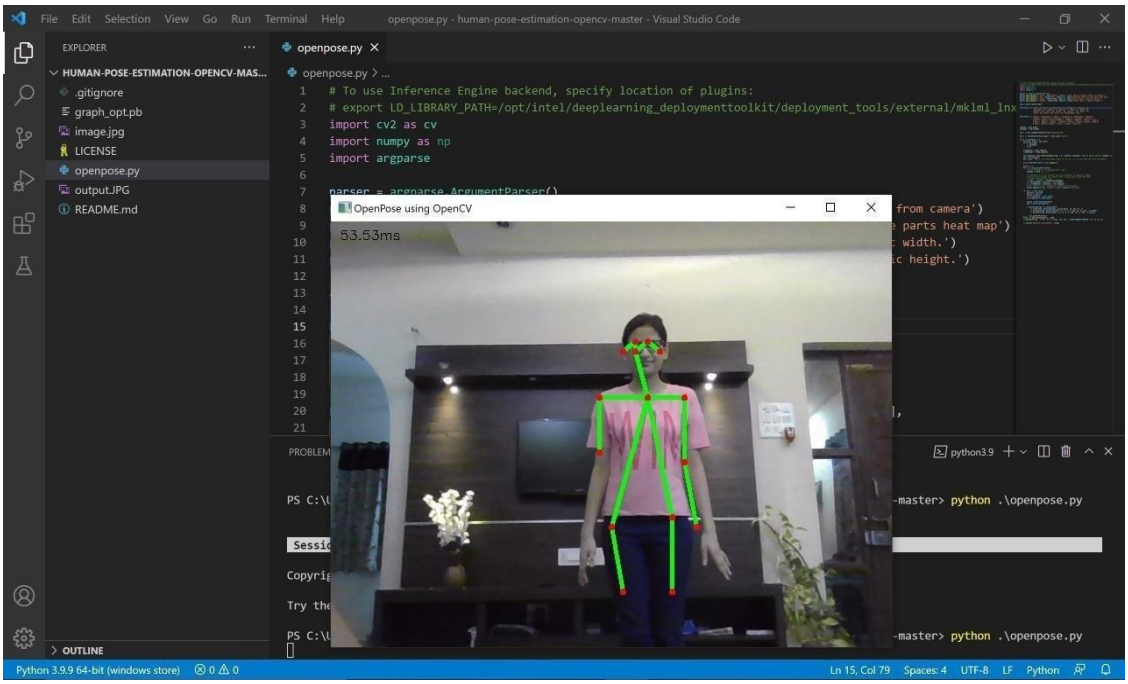
```
            circle(im, image_points[i], 3, Scalar(0,0,255), -1);
        }

        cv::line(im,image_points[0], nose_end_point2D[0],
cv::Scalar(255,0,0), 2);

        cout << "Rotation Vector " << endl << rotation_vector <<
endl;
        cout << "Translation Vector" << endl <<
translation_vector << endl;

        cout <<  nose_end_point2D << endl;

        // Display image.
        cv::imshow("Output", im);
        cv::waitKey(0);

}
```

1.5 OUTPUT:



This is the output of given input estimating the pose of a human head in a photo using OpenCV.





The Yoga application is able to capture user movements using the camera, and then check their positions or guide them accordingly.After which the algorithm will match the data with the given data set and tell the user if there are any improvements required in the position .

1.6 WORKING:

The Yoga application is able to capture user movements using the camera,and then check their positions or guide them accordingly. After which the algorithm will match the data with the given data set and tell the user if there are any improvements required in the position .The system consists of two main modules, a pose estimation module which uses OpenPose to identify keypoints in the human body, and a pose detection module which consists of a Deep Learning model, that is used in order to analyze and predict user pose or asana .This will help users to practice yoga in the correct way and get benefits for a healthy life.

Table 1.2 Keypoints used

No.	Keypoint	No.	Keypoint
0.	Nose	9.	Right knee
1.	Neck	10 .	Right foot
2.	Right shoulder	11.	Left hip
3.	Right elbow	12.	Left knee
4.	Right wrist	13.	Left foot
5.	Left shoulder	14.	Right eye
6.	Left elbow	15.	Left eye
7.	Left wrist	16.	Right ear
8.	Right hip	17.	Left ear

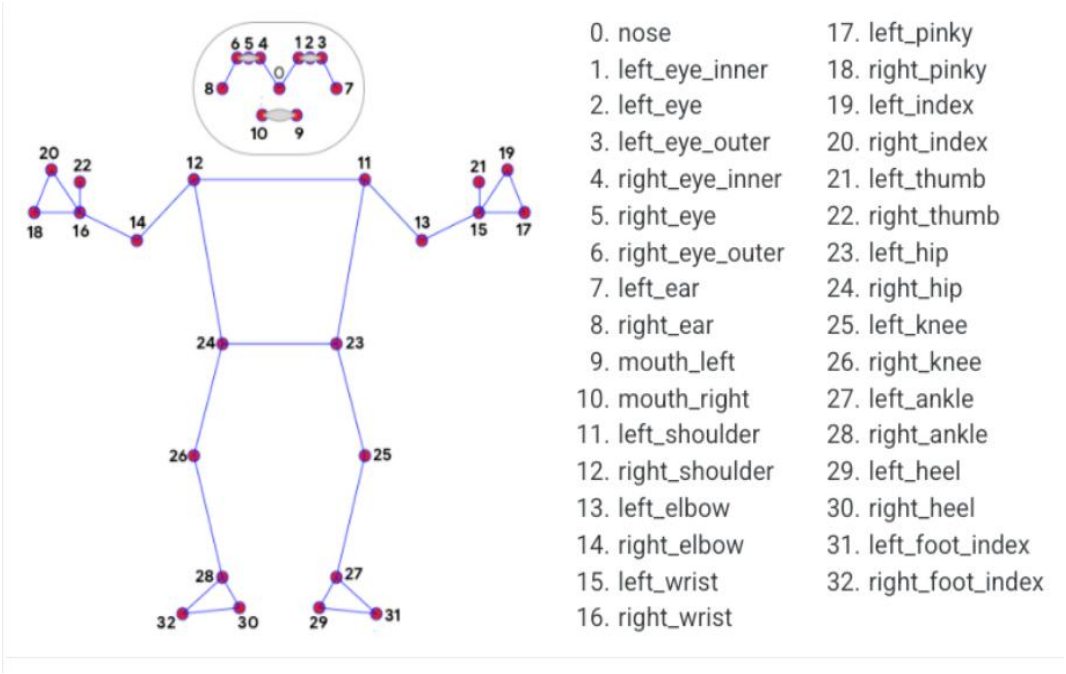


Fig 1.2 shows the This is the first step of our pipeline and the OpenPose library is utilized for it. In the case of recorded videos, this step takes place offline, whereas for real-time predictions, it takes place online using input from the camera to supply keypoints to the proposed model. OpenPose is an open source library for multi-person keypoint detection, which detects the human body, hand, and facial keypoints jointly. The positions of 18 keypoints tracked by the Open Pose, i.e. ears, eyes, nose, neck, shoulders, hips, knees, ankles, elbows, and wrists are display.

Fig 1.3 Table 3 dataset details

Sr. no.	Asana name
1	Bhujangasana
2	Padmasana
3	Shavasana
4	Tadasana
5	Trikonasana
6	Vrikshasana

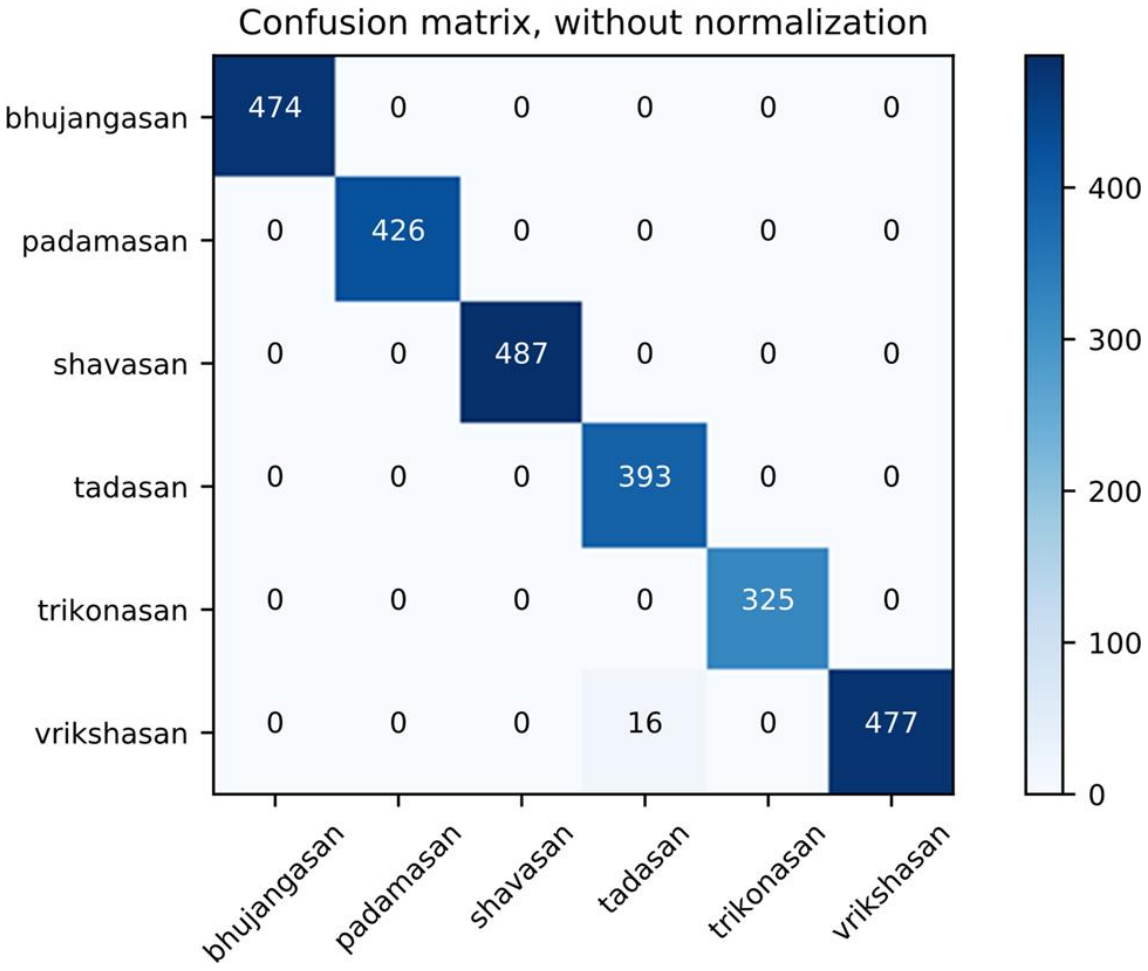


Table 4 Real-time results

S. no	Activity	Total cases	Correct cases Accuracy (%)
1	Bhujangasana	135 132	97.78
2	Padmasana	154 154	100.00
3	Shavasana	146 144	98.63
4	Tadasana	138 138	100.00
5	Trikonasana	178 178	100.00
6	Vrikshasana	178 173	

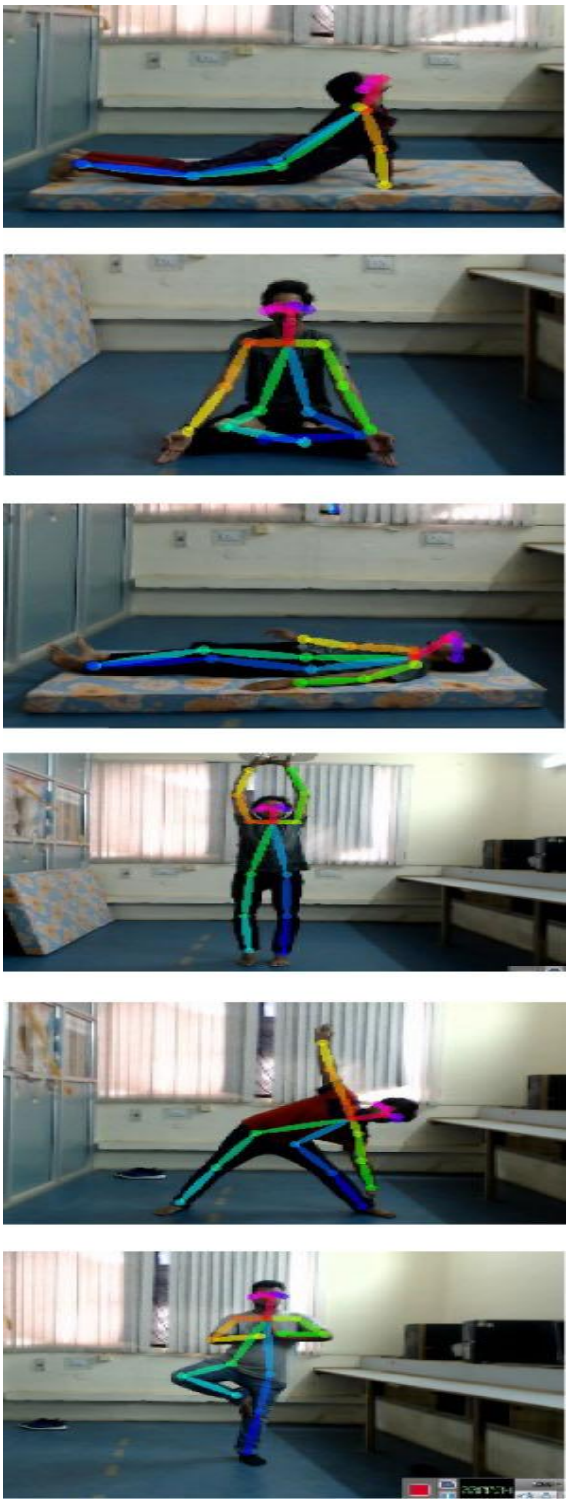


Fig.Predictions of asanas in real time (top to bottom row): Bhujangasana (column 2 has the wrong prediction), Padmasana, Shavasana, Trikonasana, and Vrikshasana.

Fig 1.3 shows The output corresponding to each frame of a video is obtained in format which contains each body part locations for every person detected in the image. The pose extraction was performed at the default resolution of OpenPose network for optimal performance. The system operated at around 3 FPS at these settings. Figure illustrates the proposed system architecture when OpenPose is used for keypoint extraction followed by the CNN and LSTM model to predict the user's asans.

1.7: CONCLUSION:

In this model Yoga identification system using a traditional RGB camera.Openpose is used to capture the user and detect the keypoints. The end to end deep learning based framework eliminates the need for making the handcrafted features allowing for addition of new asanas by just retaining the model .We applied the time -distributed CNN layer to detect patterns between key points in a single frame and the LSTM to memorize the pattern found in recent frames using LSTM for the memory of previous frames make system even more robust by minimizing error due to false keypoint detection.

2. LITERATURE REVIEW:

2.1 Estimation of yoga posture :

HISTORY OF YOGA

Humans are prone to musculoskeletal disorders with aging and accidents. To prevent this problems some form of physical exercise is needed. Yoga, which is a physical exercise, has gained tremendous significance in the community of medical researchers. Yoga has the capacity to completely cure diseases without any medicines and improve physical and mental health. vast body of literature on the medical applications of yoga has been generated which includes positive body image intervention, cardiac rehabilitation, mental illness and much more. Yoga comprises various asanas and exercises which represent physical static postures. The application of pose estimation for yoga is challenging as it involves complex configuration of postures and yoga.

HUMAN POSE ESTIMATION

Human posture recognition has made much advancements in the past years. It has developed from 2D to 3D pose estimation and from single person to multi person pose estimation.It uses pose estimation to build a machine learning application that helps detect shoplifters whereas it uses a single RGB camera to capture 3D poses of multiple people . Human pose estimation algorithms can be majorly organized in two ways. Algorithms prototyping estimation of human poses as a geometric calculation are classified as generative methods on other hand algorithms modelling human pose estimation as an image processing problem are classified as discriminative methods. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

2.2 REFERENCE:

1. Gao Z, Zhang H, Liu AA et al (2016) Human action recognition on depth dataset. Neural Comput Appl 27:2047–2054. https://doi.org/10.1007/s00521-015-2002-0

2. Poppe R (2010) A survey on vision-based human action recognition. Image Vis Comput 28:976–990. https://doi.org/10.1016/j.imavis.2009.11.014

2 3. Weinland D, Ronfard R, Boyer E (2011) A survey of vision based methods for action representation, segmentation and recognition. Comput Vis Image Underst 115:224–241. https://doi.org/10.1016/j.cviu.2010.10.002

4. Ladjailia A, Bouchrika I, Merouani HF et al (2019) Human activity recognition via optical flflow: decomposing activities into basic actions. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3951-x

5. Suto J (2018) Comparison of offlfline and real-time human activity recognition results using machine learning techniques. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3437-x

6. Guddeti RR, Dang G, Williams MA, Alla VM (2018) Role of Yoga in cardiac disease and rehabilitation. J Cardiopulm Rehabil Prev. https://doi.org/10.1097/hcr.0000000000000372

7. Neumark-Sztainer D, Watts AW, Rydell S (2018) Yoga and body image: how do young adults practicing yoga describe its impact on their body image? Body Image 27:156–168. https://doi.org/10.1016/j.bodyim.2018.09.001

8. Halliwell E, Dawson K, Burkey S (2019) A randomized experi mental evaluation of a yoga-based body image intervention. Body Image 28:119–127. https://doi.org/10.1016/j.bodyim.2018.12.005

9. Sathyanarayanan G, Vengadavaradan A, Bharadwaj B (2019) Role of yoga and mindfulness in severe mental illnesses: a narrative review. Int J Yoga 12:3–28. https://doi.org/10.4103/ijoy. IJOY_65_1

10. Patil S, Pawar A, Peshave A et al (2011) Yoga tutor: visualization and analysis using SURF algorithm. In: Proceedings of 2011 IEEE control system graduate research colloquium, ICSGRC 2011, pp 43–46

11. Chen HT, He YZ, Hsu CC et al (2014) Yoga posture recognition for self-training. In: Lecture notes in computer science (including subseries lecture notes in artifificial intelligence and lecture notes in bioinformatics), pp 496–505

12. Chen HT, He YZ, Chou CL et al (2013) Computer-assisted self training system for sports exercise using kinects. In: Electronic proceedings of 2013 IEEE international

conference multimedia and expo work ICMEW 2013 3–6. https://doi.org/10.1109/icmew.2013.6618307

13. Schure MB, Christopher J, Christopher S (2008) Mind–body medicine and the art of self-care: teaching mindfulness to coun seling students through yoga, meditation, and qigong. J CounsDev. https://doi.org/10.1002/j.1556-6678.2008.tb00625.x

14. Lim S-A, Cheong K-J (2015) Regular Yoga practice improves antioxidant status, immune function, and stress hormone releases in young healthy people: a randomized, double-blind, controlled pilot study. J Altern Complement Med 1:1. https://doi.org/10.1089/acm.2014.0044

15. Chen HT, He YZ, Hsu CC (2018) Computer-assisted yoga training system. Multimed Tools Appl 77:23969–23991. https://doi.org/10.1007/s11042-018-5721-2