Real-time Performance Evaluation of Yoga Poses Using the NVIDIA Jetson Nano: A Comparative Study Involving Stereo Vision and Body Angles Estimation Methods

by

Eric Williams-Linera
Bachelor of Science in Mechatronics Engineering
Instituto Tecnológico y de Estudios Superiores de Monterrey

A dissertation submitted in partial fulfillment of the requirements for the degree of:

Master of Science with Major in Electronics

at the

Instituto Nacional de Astrofísica, Óptica y Electrónica
September, 2024
Tonantzintla, Puebla, Mexico

Advisor:

Juan Manuel Ramírez-Cortés
Electronics Department
INAOE

"To my mom"

Contents

List of Figures

List of Tables

Acknowledgments

I thank my advisor, Dr. Juan Manuel Ramírez-Cortés, for providing me the opportunity to develop this thesis work under his supervision. Also, for the valuable suggestions, contributions and feedback that he provided throughout the development of this project.

To the members of the jury, I thank you for your support, feedback and suggestions, which were vital to produce the final and completed version of this thesis report.

I thank my mom for her words of encouragement and support over the course of these couple of years. I thank you for sharing your knowledge and expertise with me and allow me to learn from you.

I thank the National Institute of Astrophysics, Optics and Electronics (INAOE) for granting me the opportunity to be part of their community. Similarly, I thank CONAHCYT for the monetary support provided throughout the development of this thesis project.

My deepest thanks to Jomahi, Pedro and Roberto for their support, which was a vital contribution that allowed to validate the results of this work.

Abstract

Practice of Yoga has seen increased popularity in recent years as it provides several health benefits through physical, mental, and spiritual practices. While several online resources are available for people to perform yoga at home without requiring an instructor, unsupervised training can increase risk of injury, as users are not provided suggestions on how to improve. This thesis project proposes a system, hosted on an NVIDIA Jetson Nano, to evaluate user performance of yoga poses in real time. The MediaPipe Pose framework was enabled to perform pose detection to estimate the 3D location of several joints of the human body, which subsequently allowed to perform pose evaluation. Pose evaluation was performed by estimating several joint angles of a particular user and comparing them with a reference pose. The poses applied to this work were Goddess, Warrior II and Tree, which can be considered easy to perform and have been applied to other works in the literature. This work involved the implementation of both single camera and stereo vision systems, using one and two cameras, respectively, to evaluate their respective performances for human pose estimation and evaluation. Similarly, two methods to estimate body joint angles were implemented, the vector dot product and a procedure based on Inverse Kinematics. The estimated joint angles allowed to design and implement a protocol to assess users in real time while performing yoga. Through a Graphical User Interface, the proposed system provides users with a score ranging from 0 to 100, which is based on the percentage error between user angles and the reference pose. Additionally, the system enables a color scale system that visually indicates practitioners how to improve as they perform a pose. Pose detection and angle estimation results of the implemented frameworks were validated with the aid of a Kinect V1 device. Results prove that Stereo Vision outperforms the single camera system in terms of accuracy for 3D pose detection and angle estimation and is, therefore, more reliable for providing correct feedback. The proposed system will be of aid to individuals that practice yoga as it will minimize injury risk while improving physical and mental health.

Keywords: Yoga, Pose evaluation, MediaPipe, Kinect, Stereo Vision, Jetson Nano, Body Angles

Chapter 1

Introduction

Yoga is a popular and ancient discipline that combines physical, mental, and spiritual practices to provide harmony between body and mind. It offers several benefits, such as improved flexibility, muscle strength, cardiovascular health, as well as mental and psychological health [9], [10]. Recent years have seen an increment of open access online resources that have allowed individuals to practice yoga at home without requiring an instructor. However, practicing yoga without supervision can result in injury, as users are not provided with improvement suggestions. Yoga-related injuries include muscle strain and sprain, as well as general pain [11].

Numerous studies have explored systems and models for classification and identification of yoga poses [12] [13]. However, as human health and welfare have gained prominence in the field of artificial intelligence, more recent works have pivoted towards evaluation of yoga poses. Various studies, for instance, incorporate wearable devices [14], while others have implemented computer vision systems for the identification and assessment of yoga poses. Said systems make use of devices such as the Microsoft Kinect, webcams and even mobile applications [15], [16], [17], [18]. Furthermore, the incorporation of Machine Learning algorithms, as well as Deep Learning models holds promise for improving performance of these systems, potentially enabling greater accuracy and robustness [19].

In particular, Machine Learning approaches involve several algorithms, such as Random Forests (RF), Support Vector Machines (SVM) and Logistic Regression classifiers [13], [20]. Similarly, Deep Learning methods have incorporated several pre-trained Neural Networks (NNs), such as the ResNet and MobileNet, as well a custom-built Convolutional Neural Networks (CNN) [12], [21], [22], [23]. Additionally, some works have even implemented hybrid systems that combine

Neural Networks for feature extraction, which then are used to train Machine Learning models to perform classification and evaluation of Yoga [24], [25], [26].

More recently, newer approaches have incorporated human pose estimation frameworks, such as MediaPipe Pose, OpenPose and PoseNet. In general, these frameworks are capable of detecting and providing coordinates for several joints or keypoints of the body. These keypoints coordinates have allowed to design and implement systems for classification and evaluation of Yoga poses [27], [28], [29], [30].

The goal of this thesis project consisted in proposing, implementing and validating a Machine Learning-based system for evaluation of yoga poses. In terms of user engagement, the system should be user-friendly, such that individuals find it easy to interact with. Similarly, the system should provide correct and accurate feedback that is clear for the user to understand so that they can easily improve their performance.

In terms of performance, the system should be able of correctly capturing the shape of the human body, therefore, it must be accurate and robust in terms of pose estimation. In addition to pose estimation accuracy, the system must prove similar or improved performance with respect to state of the art studies. To achieve this, a validation procedure should be implemented. Additionally, the system should be able to operate in real time, while maintaining low power consumption.

The yoga poses applied to this work consisted in Goddess, Warrior II and Tree. These poses were chosen since they have already been applied to other works in the literature and are considered easy to perform. This work made use of the MediaPipe Pose Machine Learning solution, which is a state-of-the-art pipeline capable of detecting and tracking human joint locations in real time. Through MediaPipe Pose, several joint angles were proposed such that the angles of a particular user can be compared with a set of reference angles to determine the accuracy with which the user performs a particular Yoga pose.

The system was hosted on an NVIDIA Jetson Nano, a small but powerful computer, which incorporates an NVIDIA graphics processing unit (GPU) that makes it suitable for running multiple neural networks in parallel. Recent years have seen the Jetson Nano become widely popular for Artificial Intelligence and Internet of Things (IoT) applications.

1.1 Organization of the thesis

This thesis project involved several stages. Chapter 2 presents and discusses the necessary concepts that were applied to this work, such as devices and pose estimation frameworks. Similarly, an explanation is provided regarding Stereo Vision for depth estimation, as well as the implemented methods to estimate joint angles.

Chapter 3 provides an overview on several Pose Estimation frameworks. Similarly, a review is given regarding several works that focus on evaluation and assessment of yoga poses using Pose Estimation Frameworks. Chapter 4 describes the proposed methodology, which explains the computer vision system used in this work and the implemented angle estimation methods. Additionally, the chapter explains how a Kinect device was used for results validation and, lastly, the proposed pose evaluation methodology.

Chapter 5 details how a set of tests were carried out with a group of volunteers, as well as the results obtained. Results focus on the obtained angles for the yoga poses applied to this work (i.e. Goddess, Warrior II, Tree). Similarly, the complete system, powered by the Jetson Nano, is presented.

Finally, chapter 6 discusses the results obtained in this work, as well as what future work should focus on.

Chapter 2

Theoretical framework

The current chapter discusses several concepts and topics necessary to the development of the current work. Initially, an insight into the hardware used during this project is given (i.e. NVIDIA Jetson Nano and Microsoft Kinect). Additionally, the pose estimation frameworks, such as MediaPipe Pose and the Kinect's Skeletal tracker, are briefly explained. Also, a description is given regarding the Stereo Vision setup implemented for depth estimation. Finally, two methods for the estimation of angles are presented, those being the vector dot product and an inverse-kinematics-based procedure.

2.1 Hardware

2.1.1 NVIDIA Jetson Nano

NVIDIA Jetson is a family of development boards and kits produced by NVIDIA, suitable for several tasks such as Software Development, Robotics, Machine Learning and Embedded Applications. In particular, the Jetson Nano is one of the smaller members of the family, but quite powerful and suitable for Machine Learning applications [31].

With an integrated 128-core Maxwell GPU, the Jetson Nano is capable of running multiple neural networks in parallel and enable several tasks such as image classification, image segmentation, object detection and speech recognition and processing [31]. The Jetson Nano has a Quad-core ARM A57 CPU, a 4 GB 64-bit LPDDR4 memory, two MIPI CSI-2 DPHY interfaces, as well as Ethernet, HDMI, Display and several USB 3.0 ports [1]. Additionally, the Jetson Nano can be powered via Micro-USB or a 5V 4A DC power supply [32].

Figure 2.1: Jetson Nano Developer Kit [1]

2.1.2 Microsoft Kinect

The Kinect is a motion sensing device manufactured by Microsoft and presented in 2010 as an accessory for the Xbox 360. The device includes several sensors such as an RGB camera, an infrared (IR) emitter and IR depth sensor, a multi-array microphone capable of recording sound and identify the source and direction of audio waves [33]. The Kinect devices also includes a 3-axis accelerometer, as seen in Fig. 2.2. Among other characteristics, the Kinect V1 device has a 57° horizontal field of view (FOV), works at a video resolution of 30 Frames-per-Second (FPS) for the depth and color streams, and includes a tilt motor [4].

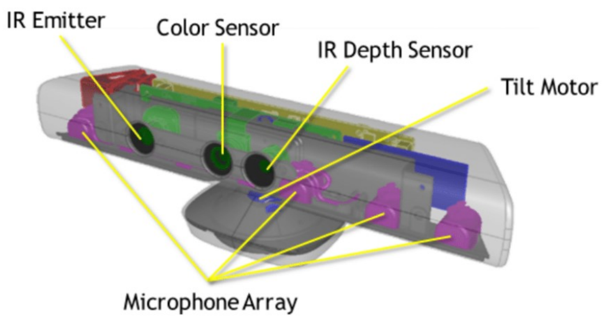
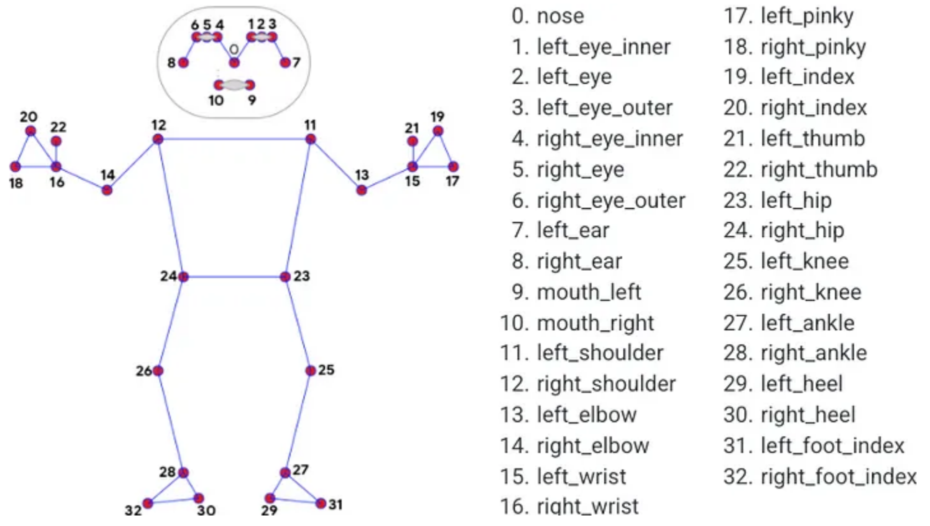


Figure 2.2: Kinect V1 schematic [Microsoft Library]

2.2 Pose estimation frameworks

2.2.1 Mediapipe Pose

MediaPipe is an open-source framework developed by Google, which allows the development of Machine-Learning-based applications [34]. Examples of use cases include pose estimation, face detection and tracking, hand gesture recognition, object detection and segmentation, as well as audio and speech recognition [35]. Specifically, the MediaPipe Pose Landmarker, which is based on Google's BlazePose model and keypoints topology, allows for real-time Pose Estimation, as it enables human body landmarks detection and tracking from both image and video sources. This framework outputs the location of 33 human body keypoints, listed in Fig. 2.3, in terms of both image pixel coordinates as well as 3D world coordinates [2].



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure 2.3: MediaPipe Pose landmarks topology [2]

The Machine Learning Solution consists of two stages, which involve a detector and a subsequent tracker model [2]. A simple process diagram can be seen in Fig. 2.4. With the detector model, the pipeline initially detects the person's region-of-interest (ROI) within the frame. Subsequently, the tracker locates the person's landmarks within the previously detected ROI [36]. For video use cases, the detector is only used when needed, such as for the very first frame and when the tracker was not able to find a body in the last frame. For the subsequent frames, the pipeline only infers the ROI from the pose landmarks inferred in the previous frame [37].

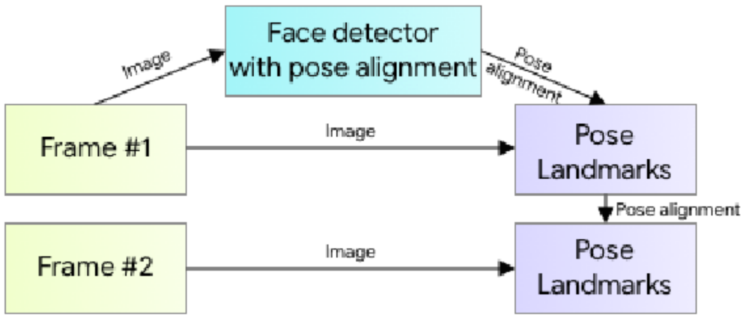


Figure 2.4: MediaPipe inference pipeline [3]

The detector model in fact makes use of the BlazeFace model, which acts as the proxy for the presence of a person. The detector model provides several parameters, such as person alignment, the middle point between the person's hip and the bounding circle circumscribing the person. The detector model then provides these parameters to the neural network that can be appreciated in Fig. 2.5 [36]. The neural network makes use of heatmap and offset loss only during training. The heatmap is discarded during inference, which was found to greatly improve the model's speed even on low-capacity devices. The neural network takes inspiration from a Stacked Hourglass architecture, but instead makes use of a small encoder-decoder heatmap network (middle, left on Fig. 2.5), and a subsequent regression encoder (right on Fig. 2.5) with several skip connections that balance high- and low-level features [37].

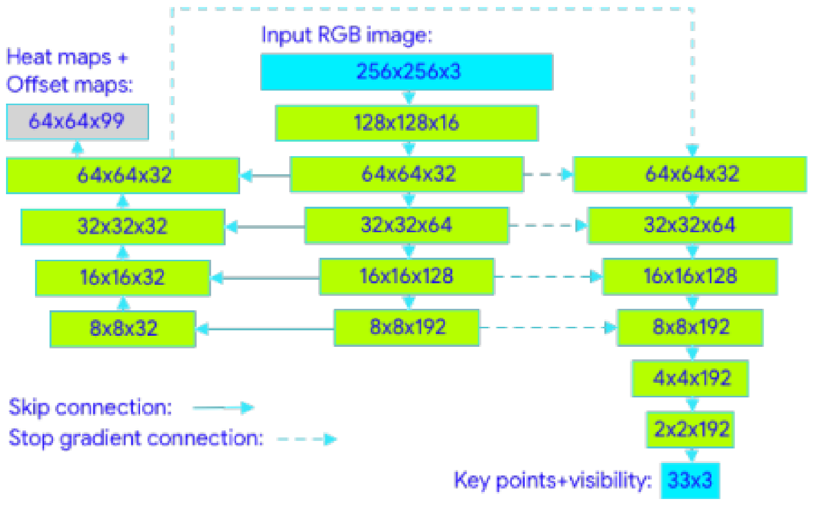


Figure 2.5: BlazePose network architecture [3]

Two BlazePose models were trained, full and lite versions, with 3.5 million and 1.3 million neural network parameters, respectively, and compared with the OpenPose Pose Estimation framework for reference. OpenPose demonstrated slightly better performance than both BlazePose models, obtaining a PCK@0.2 score of 87.8 on a custom-designed dataset by Google, while BlazePose models obtained 84.1 and 79.6, respectively. However, BlazePose models showed significantly better speed performance than OpenPose, achieving resolutions of 31 and 10 FPS, respectively, on a single core of a Pixel 2 device, compared to 0.4 FPS achieved by OpenPose on a 20-core desktop CPU (Intel i9-7900X).

2.2.2 Microsoft Kinect Skeletal Tracking

The Kinect gaming platform's pose estimation algorithm, Skeletal Tracker, was built on top of the work developed by [38] Shotton et al. The Kinect's joint detection algorithm is inspired by works based on object recognition which divide objects into parts [39] [40]. The proposed method uses single depth-images to predict 3D positions of body joints in real time and without having to rely on temporal information.

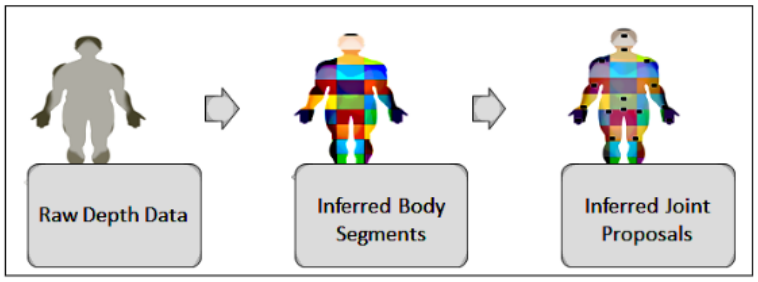


Figure 2.6: Kinect skeletal tracking process [4]

The Kinect uses an algorithm to match the incoming raw data, provided by the depth sensor, with sample training data, which is labeled such that it is associated with a particular body part. Consequently, a Decision Forest model segments the matched depth data in order to identify and propose the location for each body joint. Lastly, the 3D views of the proposed location of the body parts are calculated, which enables skeleton tracking [4]. This procedure is summarized in Fig. 2.6.

The Kinect Windows Software Development Kit (SDK) allows to access 20 skeleton joints, as well as their respective x, y and z coordinates. Fig. 2.7 illustrates the joints that are provided by the SDK.

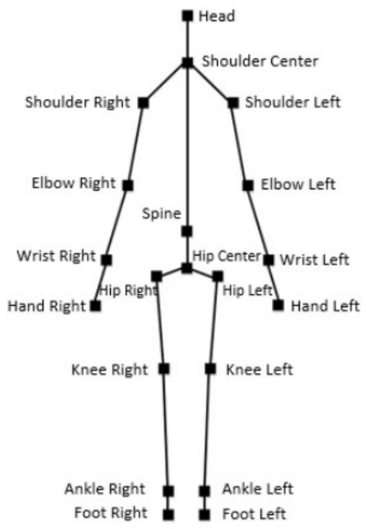


Figure 2.7: Kinect SDK skeleton tracking joints [4]

2.3 Stereo Vision

Stereo Vision systems incorporate two or more cameras oriented towards the same object [41]. The aim of a Stereo Vision System is to capture an object in the real world and estimate its depth [42]. Specifically, the goal consists in identifying the same pixel in both camera frames with the aim of obtaining a disparity between the images, which will be further used to estimate the depth of the object of interest. Fig. 2.8 depicts a simple stereo vision system [5] in which the cameras are parallel to each other. For convenience, both cameras should have the same parameters, that is, pixel resolution and focal length. As shown in Fig. 2.8, the cameras are separated by a baseline b, f represents the focal length of the cameras, P represents a point in the real world defined by (x, y, z) coordinates, and U_L and U_R represent the projections (in pixels) of point P in the left and right image frames, respectively (i.e. horizontal pixel coordinates).

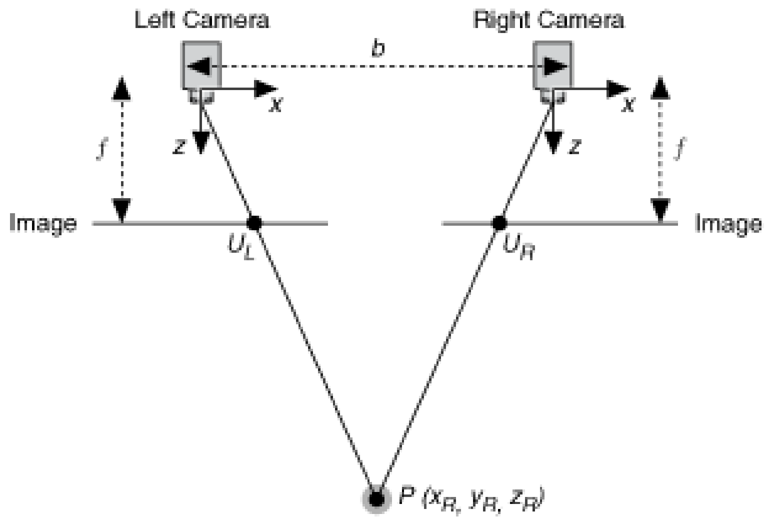


Figure 2.8: Simple Stereo Vision System. Retrieved from National Instruments [5]

It is important to mention that Fig. 2.8 assumes that the two cameras have the same focal distance, that the cameras are parallel between each other, which means there is only a horizontal disparity and not a vertical one, and that the X-axes of both cameras are collinear with the baseline axis. Equations 2.1 and 2.2 are used to obtain the projections (i.e. horizontal coordinates) for point P in the left and right camera images, respectively [41].

$$U_L = f\frac{X_A}{Z_A} \tag{2.1}$$

$$U_R = f\frac{X_A - b}{Z_A} \tag{2.2}$$

Z_A represents the optical axis of a camera, which, in this case where both cameras are parallel, represents the axis along which the depth of point P is sought to be estimated. The disparity between the projections of point P in both images frames can be calculated using Eq. 2.3.

$$d = (U_L - U_R) = f\frac{b}{Z_A} \tag{2.3}$$

From Eq. 2.3 we can obtain Z_A, that is, the depth of point P.

$$Z_A = f\frac{b}{(U_L - U_R)} \tag{2.4}$$

2.4 Angle estimation methods

2.4.1 The Vector Dot Product

Also known as the "scalar product", the vector dot product consists in a operation that is performed between two vectors of equal dimension, which yields a scalar [43]. Let u and v be two vectors of N dimensions. The dot product between u and v is defined as

$$u \cdot v = u^T v = \sum_{i=1}^{\infty} u_i v_i \tag{2.5}$$

When v is a unit vector, denoted as $\hat{v}$, performing the dot product between u and v yields the projection of u along the direction of $\hat{v}$, which means the result depends on the length of u and the cosine of the angle between u and v [44]. Let $||u||$ and $||v||$ represent the Euclidean norm of u and v, respectively; the dot product, along with the Euclidean norm, can be used to obtain the cosine of the angle between u and v

$$cos\theta = \frac{u \cdot v}{||u|| \cdot ||v||} = \hat{u} \cdot \hat{v} \tag{2.6}$$

Geometrically, the dot product represents the projection of one vector onto another, that is, how much of one vector is pointing in the direction of another vector. From Eq. 2.6, we can easily solve for the value of the angle between vectors u and v.

2.4.2 Inverse Kinematics

In the common context of robotics and even animation, Forward Kinematics involves obtaining the values for the position and movement of a robot's end-effector (e.g. gripper, tool), taking as inputs the know values of the angle or position of each individual joint [45]. It is a relatively simple task and can be solved analytically. In particular, the following transformation is set to be implemented, which converts a set of values from joint space to cartesian space

$$\theta_1\theta_2...\theta_N \rightarrow \;\; x_{EE}, y_{EE}, z_{EE} \tag{2.7}$$

Inverse Kinematics is the inverse process: given the cartesian coordinates for the position of the end-effector, the specific angles for each joint are to be obtained [46]. In this context, the following transformation is sought to be implemented

$$x_{EE}, y_{EE}, z_{EE} \rightarrow \ \theta_1\theta_2...\theta_N \tag{2.8}$$

Take for instance the robotic arm depicted in Fig. 2.9. Given that the end-effector of the robot arm has desired x and y coordinates, using an Inverse Kinematics approach based on geometry, for example, one can find the appropriate angle configuration for each joint (q_1, q_2, q_3, q_4) that would place the end-effector in the desired location.

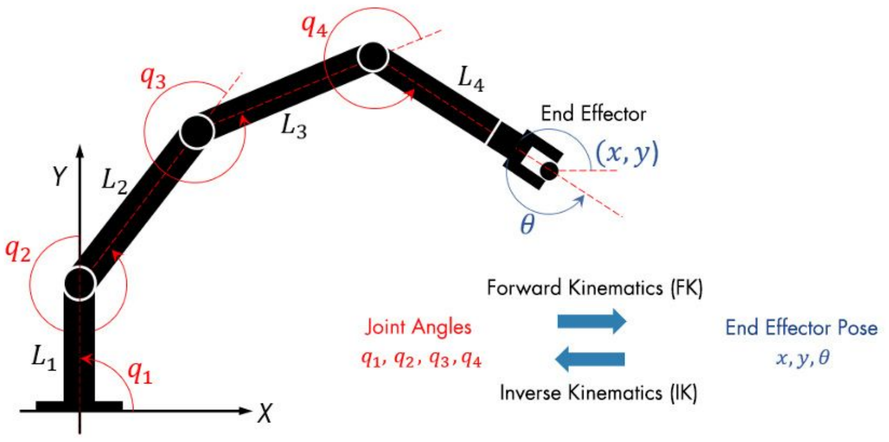


Figure 2.9: Configuration of the joint locations of a Robot Arm using Forward or Inverse Kinematics [6]

A geometric approach to solve Inverse Kinematics would consist of finding and using the geometric relationships between the links and joints of a robot arm or kinematic chain, such as the ones shown in Fig. 2.10. It can be quite straightforward and rather simple for certain types of robot manipulators, in particular simpler robots with few degrees of freedom and kinematic links. Additionally, if the coordinate values for each joint are known, this could lead to a closed-form solution, that is, an exact mathematical solution that directly calculates the joints of the robot, without needing to apply iterative methods and complex matrix operations [7]. Important geometric relationships involve the Law of Cosines and the Vector Dot Product. These can be used to obtain, for instance, the values of θ_2 and θ_3, respectively.

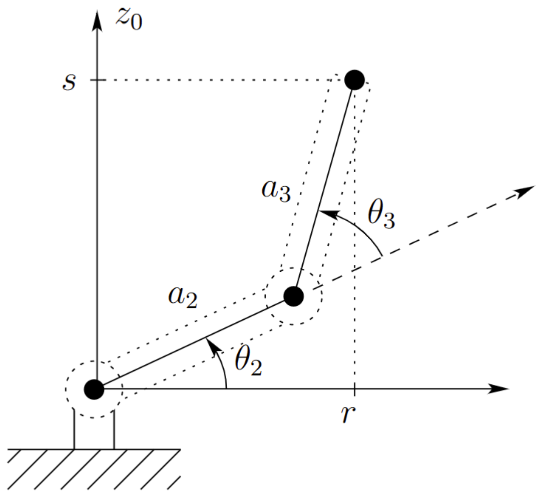


Figure 2.10: Example of a two-link manipulator with its corresponding angles [7]

Solving Inverse Kinematics using geometry offers several advantages over numerical methods, such as computational efficiency and simplicity, as it relies on geometrical relationships, rather than implementing complex and iterative mathematical and matrix computations. Additionally, geometrical approaches to solve Inverse Kinematics offer good visualization and understanding of the mechanics behind the task that is trying to be solved [47]. Similarly, geometric approaches can, in some cases, avoid singularities and indeterminate results [48].

As stated above, there is a relationship between Forward and Inverse Kinematics, which can be observed in Fig. 2.11. Forward Kinematics aims to convert from joint space to cartesian space, while Inverse Kinematics consists of the opposite procedure.

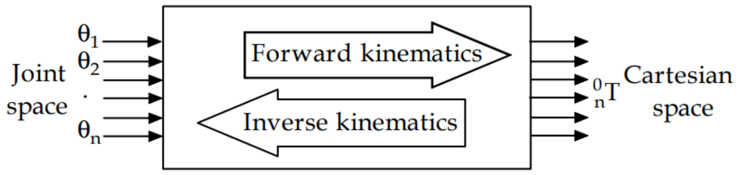


Figure 2.11: Relationship between Forward and Inverse Kinematics [8]

Inverse Kinematics, unlike Forward Kinematics, is generally not straightforward and very few analytical and numerical solutions exist. In some cases, there can be *many* solutions, or even no solutions at all [49]. In general, the solution of an inverse kinematics problem can be computationally expensive and require long time. Moreover, the problem's complexity increases due to singularities and nonlinearities [50]. Therefore, analytical solutions only exist for a specific set of scenarios.

Chapter 3

Related work

There are several works in the literature that implement algorithms and frameworks for the identification and classification of yoga poses, mainly through Convolutional Neural Networks (CNNs) and classification models such as Support Vector Machines (SVMs) and Decision Trees [12] [13]. More recently, focus has shifted towards the development of yoga posture correction and evaluation systems. This chapter makes an overview on several Pose Estimation frameworks that have been applied to develop systems to evaluate yoga performance.

In subsection 2.2.1 it was discussed how MediaPipe Pose consists of two stages, in which the first stage consists of a detector that finds the Region-of-Interest (ROI) of an individual, along with a few keypoints, while the second stage involves a tracker that completes the mapping of the pose. Besides MediaPipe Pose, there are several other Pose Estimation frameworks that have been used for pose evaluation, such as OpenPose, PoseNet and MoveNet. In general, these models have become widely popular for particular applications but differ in their internal structure and operation. Their respective architecture makes each of them suitable for different contexts as they offer several advantages and drawbacks.

Launched and enhanced during 2017, OpenPose is a multi-person framework that can detect up to 135 keypoints, including body, hand, facial and foot, from single images [51]. OpenPose makes use of several steps to perform Pose Estimation. Initially, the input image is analyzed by Convolutional Neural Network (CNN) to extract feature maps. The initial CNN consists of the first 10 layers of the VGG-19 network. The extracted feature maps are then processed by a multi-stage CNN that generates Part Confidence Maps and Part Affinity fields, which help to identify the possible locations of a particular body part, and its orientation, respectively. In the last step, the generated Part Confidence Maps and Part Affinity Fields are processed with a bipartite matching algorithm that produces the final pose for each person in the image [52].

The TensorFlow.js version of PoseNet was released by Google Creative Lab in 2018. It allows for single and multi-person pose estimation from image and video sources. For the single pose version, the PoseNet model resizes the input frame with the aim of obtaining either higher accuracy or runtime performance. PoseNet processes the image with the aid of a pre-trained MobileNet model that outputs heatmaps and offset vectors. These are used to obtain confidence locations for each keypoint in the frame. Lastly, several calculations are performed on the obtained heatmaps and confidence vectors that return the most likely location for each keypoint [53].

MoveNet was also developed by Google and it consists of an ultra fast model capable of detecting 17 keypoints. The MoveNet architecture makes use of heatmaps to find the locations of human joints. The model initially employs a feature extractor model, based on the MobileNet-V2. Then, it makes use of a set of prediction heads that analyze the extracted features and estimate several features such as the person center heatmap and person keypoints. Due to not making use of a detector to identify persons in image frames, MoveNet directly predicts the location of the keypoints, which allows for greater speed while potentially sacrificing accuracy [54].

Several datasets exist to determine the estimation accuracy of a particular Pose Estimation model. Among the most popular, there is the Common Objects in Context (COCO) [55] and MPII [56], along with others, such as HIIT. In general, these datasets contain images of people performing several activities in different environments and contexts. These images contain annotations regarding the location of body joints or keypoints in terms of the image pixels. One particular metric to evaluate the accuracy and performance of a given model is the mean Average Precision (mAP), which summarizes the overall performance of an object detection model across several categories within a specific dataset. It is calculated by averaging all Average Precision (AP) values obtained for each object class within a dataset [57]. While the Average Precision (AP) metric focuses on a particular class of a dataset, the mean Average Precision (mAP) provides an overview of the model's performance across all of the dataset's categories.

Table 3.1 represents a brief performance comparison between the aforementioned Pose Esimation benchmarks, highlighting whether they are single or multi-pose, the number of keypoints they can detect, as well as their coordinates. Additionally, their speed in frames-per-second (FPS) and estimation accuracy on a particular dataset is provided. In this context, the COCO dataset makes no distinction between Average Precision (AP) and mean Average Precision (mAP).

Framework	Number of poses	Number of keypoints	Speed [FPS]	Accuracy [mAP, dataset]
MediaPipe Pose [58]	Single pose	33, 3D	20-50 FPS (Pixel 3 GPU)	74.0, HIIT
OpenPose [52]	Multi-pose	Up to 135, 2D and 3D	22 FPS (NVIDIA GTX 1080 Ti)	75.6, MPII
PoseNet [53]	Single and multi-pose	17, 2D	25 FPS (Pixel 5 GPU)	0.687 (Average Precision), COCO 2016
MoveNet (Lightning) [59]	Single and multi-pose	17, 2D	40 FPS (Pixel 5 GPU)	66.8, COCO 2017 [60]
MoveNet (Thunder) [59]	Single pose	17, 2D	22 FPS (Pixel 5 GPU)	77.8, COCO 2017 [60]

Table 3.1: Performance comparison between several Pose Estimation benchmarks

Table 3.2 outlines several works that have made use of Pose Estimation frameworks to develop Yoga pose evaluation systems, mainly MediaPipe Pose, OpenPose, PoseNet and MoveNet. The table includes columns for the pose estimation framework and computer vision system used, the method to perform pose evaluation, as well as how feedback is provided to the user. The table also specifies the platform on which the system was hosted on.

Year	Pose estimation framework	Pose evaluation method	Feedback form	Platform	Ref
2019	OpenPose, Single webcam	Joint angles using OpenPose keypoints	Red-green color scale, "Perfect" to "Bad" scale		[15]
2021	OpenPose, Single webcam	Joint angles and distances	Step by step instructions	4 GB RAM machine	[16]
2020	OpenPose, Phone camera	Machine Learning Classification Model	Confidence value of ML Model	Android App	[17]
2022	MediaPipe Pose, Phone camera	Joint angles using MediaPipe keypoints	Step by step instructions via Google Text-to-speech API	Android App	[28]
2024	PoseNet, Single webcam	Joint angles using PoseNet keypoints	Text, audio messages to correct posture	Web App	[29]
2023	MediaPipe Pose Single webcam	Body angles using MediaPipe keypoints	Success/negative message	64-bit OS	[61]
2024	MediaPipe Pose, Single webcam	Joint angles using MediaPipe keypoints	Suggestions provided by Neural Network		[62]
2020	PoseNet, Single webcam	Joint angles using PoseNet keypoints, ML Model for feedback	Correct/Wrong pose message, more/less/ok message for each joint		[63]
2020	PoseNet, phone camera	Joint angles and distances	"Pose correctly performed" message	Android App	[64]
2023	PoseNet, Single webcam	Joint angles using PoseNet keypoints	Green/white skeleton drawing for correct/incorrect pose	Web App	[65]
2024	MoveNet, Single webcam	Joint angles using MoveNet keypoints	Green/white skeleton drawing for correct/incorrect pose	Web App	[66]

Table 3.2: Similar works that have developed Yoga evaluation systems using Pose Estimation frameworks

Chapter 4

Proposed methodology

The current chapter discusses the proposed methodology for this work. Initially, an explanation of the software and hardware behind pose estimation and keypoints detection is presented. Secondly, the angles that were proposed to define the human body, as well as the mathematical procedures to estimate them, are explained. Also, the evaluation and feedback protocol to determine the accuracy of a given pose, are described. Lastly, validation of angle estimation via a Kinect V1 device is also explained. A flowchart of the overall methodology is presented towards the end of the chapter.

4.1 System block diagram

The proposed system incorporates IMX219 cameras connected to the Jetson Nano via the CSI interfaces. The system takes as input an image or pair of images. MediaPipe Pose is then enabled to perform pose estimation, which provides the keypoints coordinates for several body joints. With these coordinates, several body joints are estimated and then compared with the reference angles of the pose the user is performing. A score, ranging from 0 to 100 is provided, as well as a color scale system to visually indicate the user which joints they could improve. The system's diagram can be seen in Fig. 4.1.

4.2 Hardware for Computer Vision

Two systems for computer vision were implemented in this work. The first one consisting of a single IMX219 camera, while the second one incorporated two IMX219 cameras, comprising a Stereo Vision System. Both IMX219 cameras, acquired from Waveshare, have an 8MP resolution and

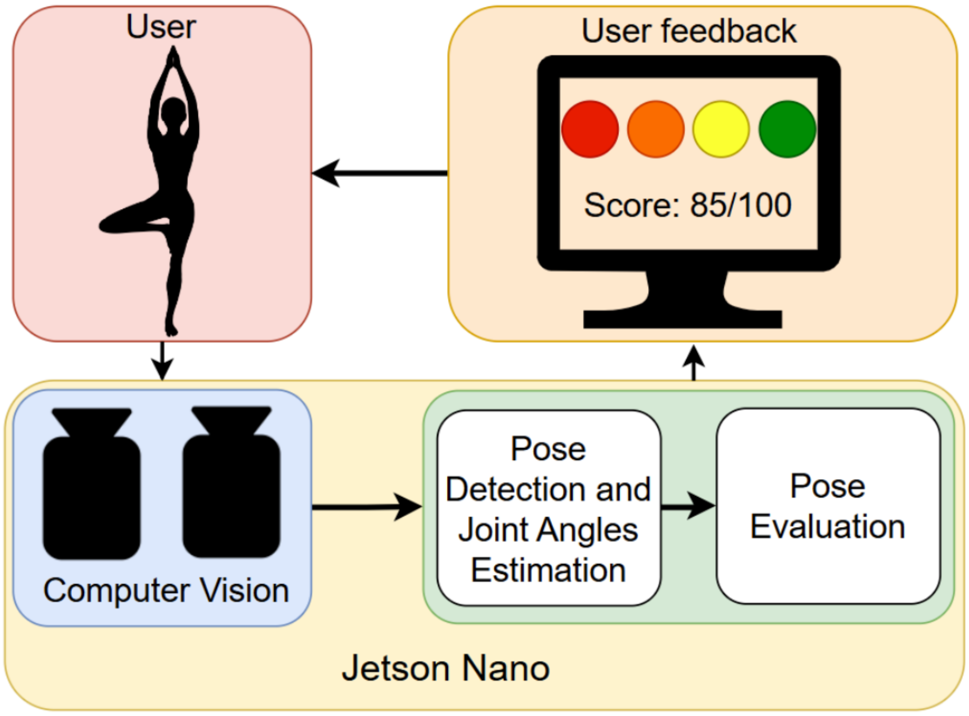


Figure 4.1: Block diagram of the yoga pose evaluation system

incorporate an IMX219 CMOS sensor, manufactured by Sony [67]. Incorporating CSI serial output, they are compatible with several boards such as Jetson and Raspberry devices. In particular, the Waveshare cameras incorporated to this work have a diagonal field of view (FOV) of 160°, a focal length of 3.15 mm and a 6.5mm x 6.5mm lens [68]. The goal of implementing two computer vision systems consisted in evaluating their respective performance and accuracy for 3D pose detection and angle estimation. Both computer vision systems were hosted on an NVIDIA Jetson Nano. The final assembly can be seen in Fig. 4.2.

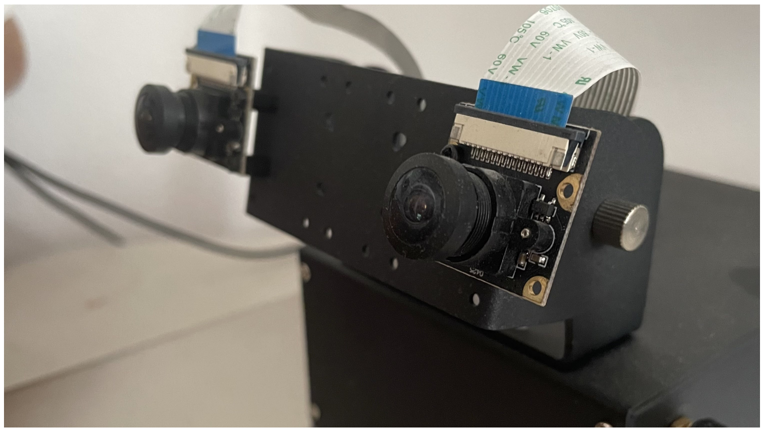


Figure 4.2: IMX219 cameras hosted on the Jetson Nano

4.3 Framework for Pose Detection

The MediaPipe Pose framework was chosen to enable pose estimation using the Jetson Nano. As it was discussed in 2.2.1, MediaPipe Pose can provide x, y and z keypoints coordinates for up to 33 keypoints of the human body. Additionally, as it was summarized in Table 3.1, MediaPipe Pose is capable of running at a superior frame rate compared to other pose estimation frameworks, making it suitable to be incorporated to a real-time pose evaluation application. While MediaPipe Pose may not be as accurate as OpenPose and the Thunder version of MoveNet, its lightweight and efficient design make it the ideal platform for this project.

When working with a single IMX219 camera, all three coordinate values provided by MediaPipe were made use of to perform pose evaluation. When using the Stereo Vision System, MediaPipe Pose was only tasked with providing x and y landmark coordinates while the Stereo Vision System, comprised of both IMX219 cameras, was used to perform depth estimation to provide the z coordinate (in cm), instead of using the one generated by MediaPipe Pose.

4.4 Methods for body angles estimation

This work proposed making use of the keypoints provided by MediaPipe, as well as the estimated depth values via Stereo Vision, to calculate several joint angles of the human body. These angles were then used to compare the pose of a user with a set of predefined angles acting as reference, such as those of a certified yoga instructor. The following subsections discuss two methods to estimate body angles, the vector dot product, and a procedure based on Inverse Kinematics.

4.4.1 Dot product

Previously, it was described how the dot product can be used to obtain the angle between two n-dimensional vectors. For this work, the dot product formula was used to obtain the angles between several links of the human body, which were treated as 3-dimensional vectors. This was achieved using the following formula, which solves for the cosine of the angle between vectors u and v

$$cos\,\theta = \frac{u \cdot v}{||u|| \cdot ||v||} \tag{4.1}$$

Fig. 4.3 depicts how Eq. 4.1 was used. A, B and C represent particular joints of the human body, whose 3-dimensional coordinates were provided by the previously described pose estimation frameworks.

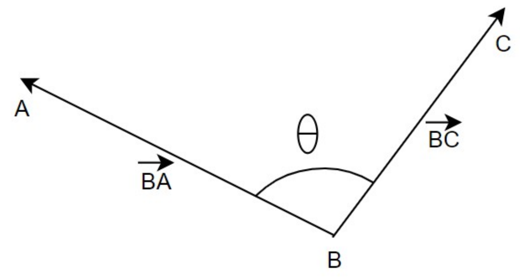


Figure 4.3: Intended use of the dot product formula

Table 4.1 lists all angles that were implemented using this methodology. And Fig. 4.4 portrays the same angles.

	Joint		Joint
1	Left Shoulder	7	Left Knee
2	Right Shoulder	8	Right Knee
3	Left Elbow	9	Left Shoulder 2
4	Right Elbow	10	Right Shoulder 2
5	Left Hip	11	Left Hip 2
6	Right Hip	12	Right Hip 2

Table 4.1: List of angles implemented using the dot product procedure

Note that in Fig. 4.4, only half of the angles are shown (e.g. Left Shoulder (1) is depicted while Right Shoulder (2) is not, Right Elbow (4) is shown while Left Elbow (3) is not)

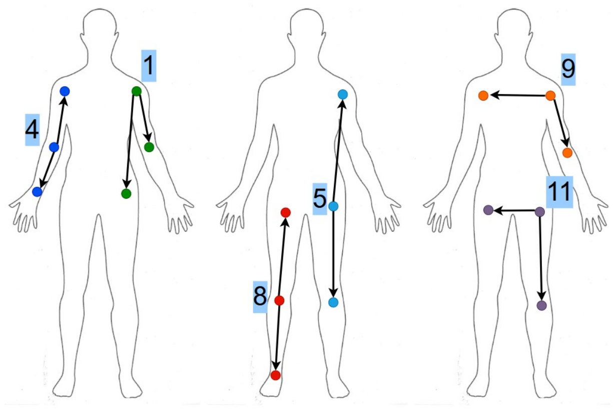


Figure 4.4: Dot Product angles drawn on the human body

4.4.2 Inverse kinematics

As it was previously described, inverse kinematics consists in obtaining the angle configuration that would yield the cartesian coordinates of the end effector of a kinematic chain (e.g. robotic arm, animated character's limb). For a geometry-based approach, the process could begin by obtaining the angle of the first link of the kinematic chain with respect to a reference plane (e.g. ground plane to which a robotic arm is attached). Then, it is necessary to obtain the angle of the second link with respect to the projection of the first link. This second step can then be repeated until the end effector (the end of the kinematic chain) is reached.

In order to apply the previous methodology to this work, it was first necessary to define two components:

1. The reference plane
2. The kinematic chains of the human body

The coronal or frontal plane, which is the plane that divides the body into front and back sections, as seen in Fig. 4.5, was proposed to serve as the reference frame for all the kinematic chains subsequently defined. A plane can be defined by three points; in this case, it was proposed that the plane was defined by the coordinates of the left shoulder, right shoulder and center hip. MediaPipe Pose does not provide coordinates for the center hip, therefore, the left and right hip coordinate values were averaged to obtain an approximation of the center hip.

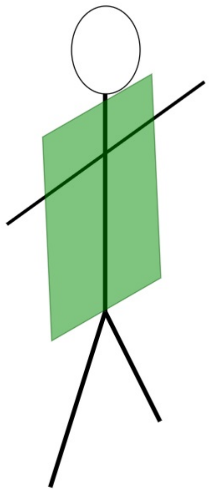


Figure 4.5: Frontal plane (green) of the human body to serve as reference plane

The next step consisted in defining all the kinematic chains that will compose the human body. This work proposed using both arms and both legs. For the case of the arms, the chain would begin with a link ranging from the shoulder to the elbow, and a second link going from the elbow to the wrist. On the other hand, for the legs, the kinematic chain would begin with a link going from a hip (e.g. left, right) to a knee, and a second link ranging from the knee to the ankle. The above described definition yields four kinematic chains overall. The reference plane, as well as kinematic chains, are illustrated in Fig. 4.6.

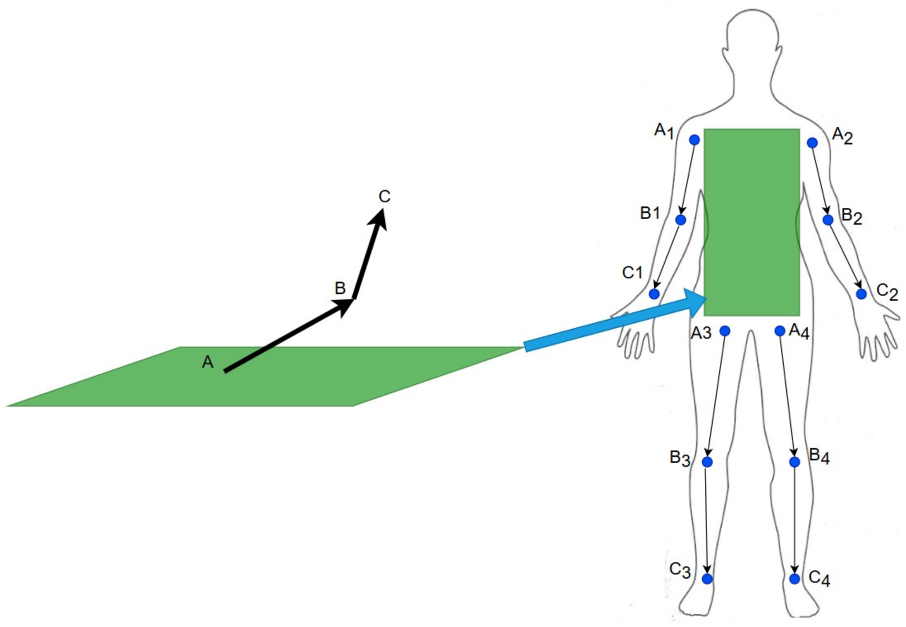


Figure 4.6: Proposed reference plane and kinematic chains

Once both the reference plane and the kinematic chains were defined, we can describe the procedure to obtain the body angles based on the inverse kinematics methodology. Initially, it was necessary to obtain the angle between a particular kinematic chain link and the reference plane. An example is provided in Fig. 4.7.

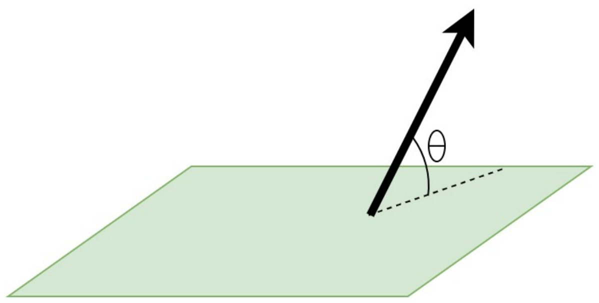


Figure 4.7: Example of angle with respect to reference plane

In order to obtain the angle between a particular link, that is, a vector defined by two points, and the reference plane, we can make use of the dot product formula between the kinematic chain's link and a vector perpendicular to the reference plane. We can then solve for the cosine of the angle between both vectors. Normally, we would solve for the angle by taking the inverse cosine to the result, however, we can instead apply the inverse sine, since we are trying to obtain the angle with respect to the plane. In order to obtain the vector perpendicular to the plane, we simply need to perform the cross product between two vectors within the plane. To achieve that, we can use the aforementioned points, i.e. left shoulder, right shoulder and center hip, to obtain two vectors and compute the cross product.

In mathematical terms, the procedure is as follows. First, two vectors within the reference plane are defined using the coordinates of the left shoulder, right shoulder and center hip. Both vectors will have their origin at the hip. Let $P1$ represent the left shoulder, $P2$ the center hip and $P3$ the right shoulder, the vectors within the plane are defined as

$$v1 = \{P2_x - P1_x; P2_y - P1_y; P2_z - P1_z\} \tag{4.2}$$

$$v2 = \{P2_x - P3_x; P2_y - P3_y; P2_z - P3_z\} \tag{4.3}$$

Once $v1$ and $v2$ have been defined, we can compute the cross product between them to obtain a vector perpendicular to the plane. This yields a vector vp

$$vp = \{A, B, C\} \tag{4.4}$$

Let L represent a particular kinematic chain link, that is, a vector with coefficients

$$L = \{l, m, n\} \tag{4.5}$$

We can make use of the dot product formula, solve for the the angle between vp and L but instead apply the inverse sine to the result to obtain the angle between L and the reference plane

$$\sin\rho = \frac{|A \cdot l + B \cdot m + C \cdot n|}{\sqrt{A^2 + B^2 + C^2}\sqrt{l^2 + m^2 + n^2}} \tag{4.6}$$

As it was described earlier, the inverse kinematic methodology implemented in this work made use of four kinematic chains, each consisting of two links. For the first link of each kinematic chain (i.e. the link attached to the reference plane), the angle that they projected onto the reference plane was obtained, this provided four initial angles. Similarly, the same process was repeated for the second link for each of the four kinematic chains, providing four additional angles. This process is illustrated in Fig. 4.8.

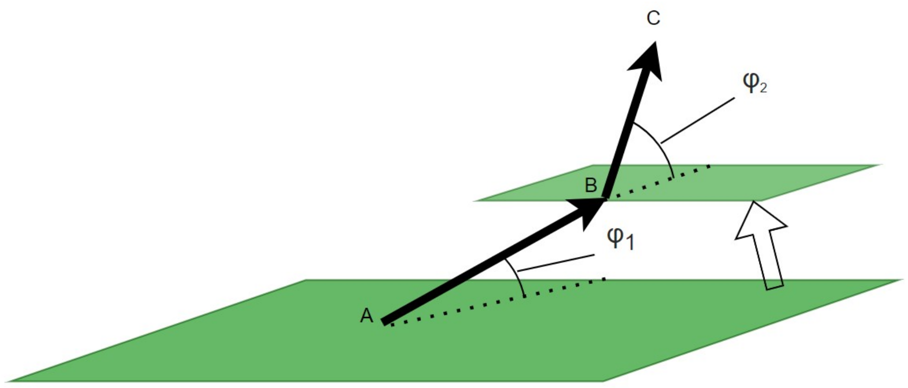


Figure 4.8: Angle with respect to plane for each link

Additionally, for each kinematic chain, the angle that the second link projects onto the first link was obtained. This was achieved by making use of the dot product formula. This is shown in Fig. 4.9. This procedure provided four additional angles, giving a total 12 angles for the inverse-kinematics-based procedure.

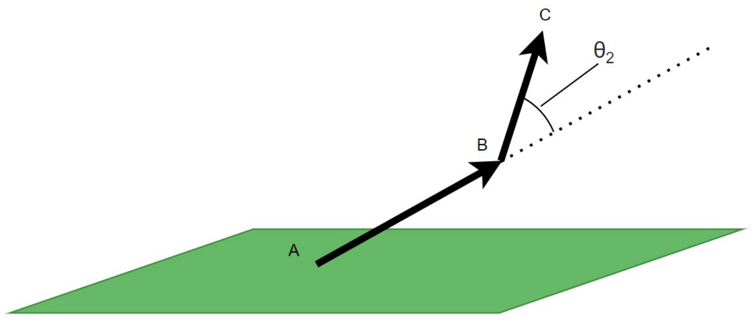


Figure 4.9: Additional angles for the inverse kinematics methodology

4.5 Validation of Angles Estimation with a Kinect V1 Device

A Kinect V1 device was enabled to validate the results obtained by both the single camera and Stereo Vision systems. The justification of using the Kinect was that it consists of a patented product and, at some point, was considered state of the art in terms of pose estimation. The Kinect was programmed to perform angle estimation using both previously described methodologies. It is important to mention that the Kinect, single camera, and Stereo Vision systems operated simultaneously but independently while estimating joint angles during testing with volunteers.

4.6 Proposed pose evaluation protocol

After enabling both pose detection, as well as angle estimation, it was necessary to design a protocol to determine the accuracy with which a user performs a certain pose. A point-scoring system was proposed in which the user is awarded points as they are performing a pose. The flowchart of the evaluation protocol is presented in Fig. 4.10.

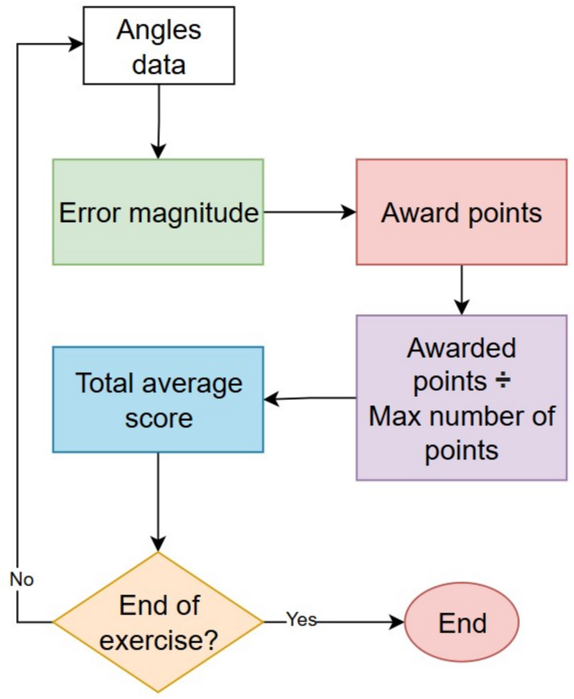


Figure 4.10: Flowchart of the point-scoring protocol

Each estimated angle is compared with the reference angle, such that a number of points is awarded, which is a function of the magnitude of the percent error between the estimated and the reference angle. Once all angles are compared, the total awarded points is divided by the maximum amount of points the user could have obtained and a total score is obtained. Also, it was proposed the user should be able to choose between several difficulty levels, where each level would mean a different complexity in terms of correctly performing a pose.

4.6.1 Definition of Reference Angles for Each Yoga Pose

In order to obtain the values of the reference angles for each pose, several images were downloaded from an open access dataset available online, on the Kaggle platform. The dataset contained several images of people performing the Yoga poses applied to this work, such as Goddess, Warrior II and Tree. With the retrieved images, both proposed angle estimation methods (i.e. Vector Dot Product and Inverse Kinematics) were applied to all images using the MediaPipe Pose framework. It is important to mention that for this procedure, no depth estimation was implemented via the Stereo System, nor was the Kinect device enabled, as all keypoints coordinates values provided by MediaPipe Pose were used. Once all images for a particular pose were processed with MediaPipe Pose, several values for the angles were obtained. These values were then averaged in order to obtain the reference angles for each pose. The angles obtained were validated by visual inspection to verify they were accurate for each joint and pose.

4.6.2 Score System for Pose Evaluation

As it was mentioned earlier, in order to determine the accuracy with which a user is performing a particular pose, a certain amount of points is awarded. This amount depends on the magnitude of the error between the angles of the user and the previously described reference angles. Eq. 4.7 was used to obtain the value of the percentage error, where v_E represents the estimated angle and v_R stands for its corresponding reference angle.

$$\%Error = \frac{v_E - v_R}{v_R} \cdot 100\% \tag{4.7}$$

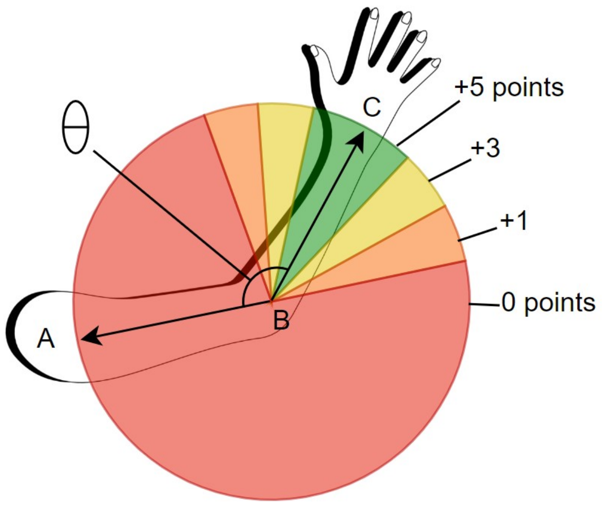


Figure 4.11: Depiction of the point-scoring system [Arm drawing image taken from Vecteezy]

An example is provided in Fig. 4.11 in which the measured angle (θ) subjected to evaluation is the one that represents the elbow. If the error is relatively small, the user is awarded the highest possible amount of points. The amount of points awarded reduces as the percentage error increases, that is, if the real angle being performed by the user differs significantly from the reference value. This process is performed for each and every one of the angles that were previously defined and for both procedures, dot product and inverse kinematics. Furthermore, three difficulty levels were defined, being easy, intermediate and hard. As the difficulty level increases, the error tolerance decreases. Table 4.2 summarizes the error value thresholds and the amount of points awarded, as a function of the difficulty level.

Easy		Medium		Hard	
Error magnitude	Points awarded	Error magnitude	Points awarded	Error magnitude	Points awarded
20 or lower	5	10 or lower	5	5 or lower	5
20 to 35	3	10 to 25	3	5 to 15	3
35 to 60	1	25 to 40	1	15 to 30	1
60 or higher	0	40 or higher	0	30 or higher	0

Table 4.2: Points system as a function of difficulty level

As the error increases, the points obtained decreases. Once the previous procedure has been applied to all angles for a particular image frame, a global score is calculated dividing the amount of points the user was awarded by the hypothetical maximum amount of points they could have obtained. Since this whole procedure is continuously being carried out, the global score value is constantly being recorded (once for every time an image frame is processed) such that the overall score is the average of all the recorded values for the global score.

4.6.3 Color Scale System for Feedback

Following the procedure of Fig. 4.11, a color scale system was implemented in which colored circles are drawn on top of a stick figure, which mimics the pose of the user. The goal of implementing the color scale is to visually indicate the user which joints are being correctly performed and which should be corrected, and how much. The color of a particular circle also depends on the magnitude of the percentage error. The stick figure is drawn using the x and y keypoints coordinates provided by MediaPipe Pose and is displayed next to the live video feed on the graphical user interface, which will be presented ahead. An example of the stick figure is shown in Fig. 4.12.

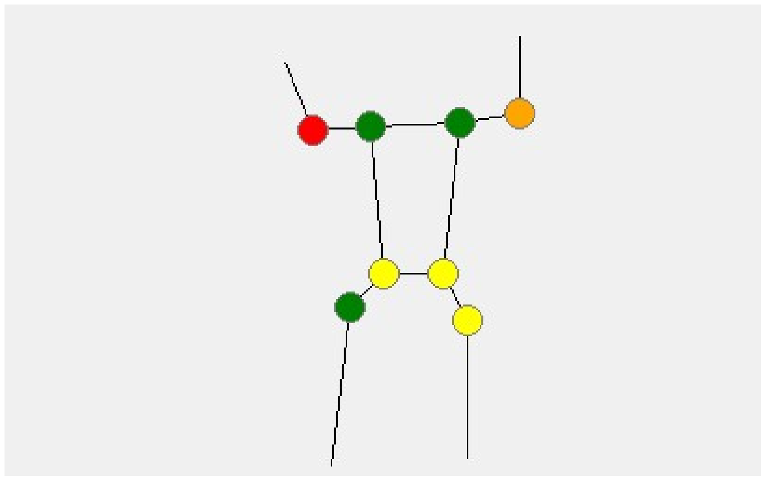


Figure 4.12: Example of the stick figure and color scale system

4.7 System flowchart

Fig. 4.13 depicts the flow diagram of the complete methodology. To achieve higher video feed FPS resolution, two threads were implemented such that they ran concurrently. This was achieved via the *threading* library available in Python. Thread 1 displays the live video feed from either of the IMX219 cameras, while Thread 2 periodically retrieves a frame, or pair of frames, from Thread 1 to perform pose evaluation (i.e. Fig. 4.10). Once the yoga exercise is completed, both threads are terminated.

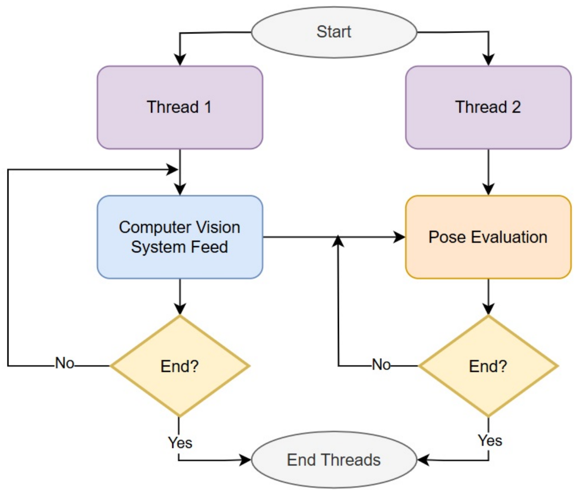


Figure 4.13: Flowchart of the complete methodology process

Chapter 5

Results

The current chapter presents the results obtained for this work. Initially, an explanation is given regarding how tests with a set of volunteers were carried out. The goal of the tests consisted of both estimating angles of several people of different body shapes, as well as evaluating the estimation accuracy of both computer vision systems (i.e. single camera and Stereo Vision) using both angle estimation methods (i.e. Dot Product and Inverse Kinematics) and also comparing them with the reference angles that were obtained for each pose. The conducted tests allowed to obtain several plots of the angles that were obtained making use of the three computer vision systems and the two implemented angle estimation methods. It is important to restate that the Microsoft Kinect's role was to act as the ground truth for the estimations obtained with the single camera and Stereo Vision systems. After the tests section is explained, the results obtained using each angle estimation are shown, first, using the Dot Product, and then via Inverse Kinematics. Additionally, a numerical comparison is presented, in the form of a table, in which the obtained results are expressed in terms of the root mean square error (RMSE) with respect to the Kinect. Lastly, the designed Graphical User Interface (GUI) is presented.

5.1 Tests with a group of volunteers

A set of tests was carried out with four volunteers. Each volunteer was asked to perform each pose (i.e. Goddess, Warrior II and Tree) once for a few seconds while both the Kinect, as well as the Jetson Nano were enabled and recording live data. The Kinect was programmed to estimate angles using both methods and record the obtained values on a .csv file. Meanwhile, the Jetson Nano was configured to save pictures, using both cameras connected to it, every time a key was pressed.

This means that every time the "save" key was pressed, two images were recorded. These images were then processed using both angle estimation methods and the remaining two computer vision systems (Stereo Vision, using two images, and single webcam, using one of the images only).

5.2 Dot Product Results

This section of the chapter presents the results of the estimated angles using the Vector Dot Product method. Initially, the results for the Goddess pose are presented, followed by the Warrior II pose, as well as the Tree pose.

5.2.1 Goddess Pose

A test example is shown on Fig. 5.1 where a particular volunteer is performing the Goddess Pose while both the Kinect as well as the Jetson Nano are recording data.

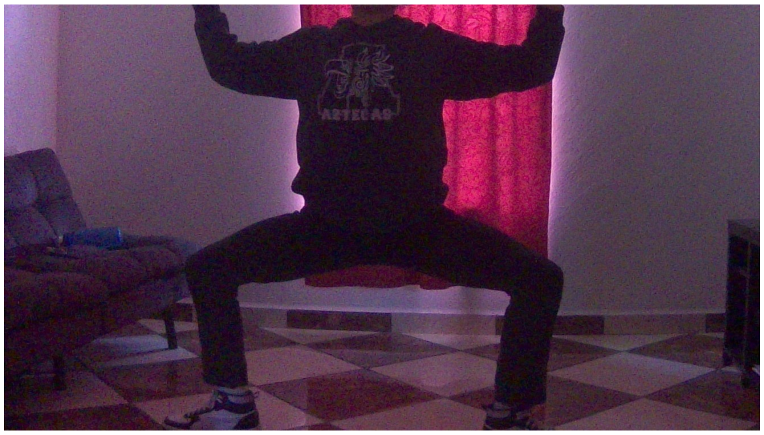


Figure 5.1: First volunteer performing Goddess Pose

The results of the estimated angles for the Goddess Pose are presented in the form a radar chart, in Fig. 5.2, where the chart axis represents the value of the angle in degrees and each variable represents a particular joint. For this figure in particular, the data represents the results obtained using the Kinect device

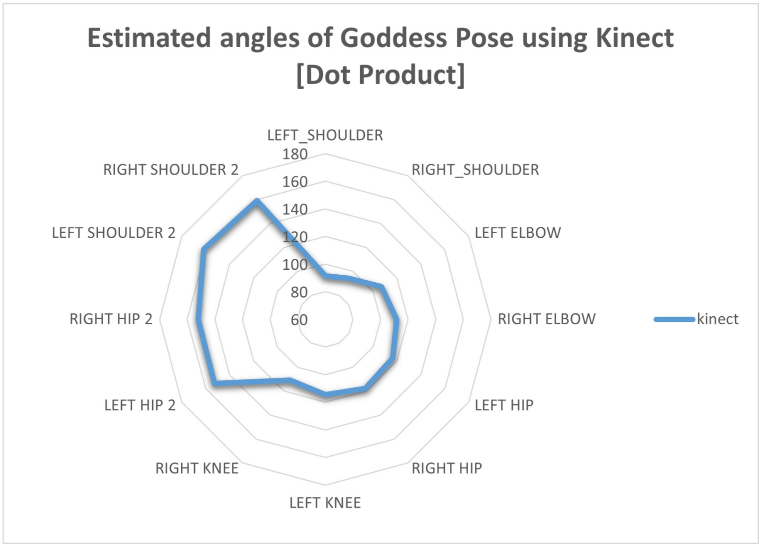


Figure 5.2: Kinect results of Goddess Pose, using Dot Product

Similarly, it was possible to obtain the results of the estimated angles, using the same chart type, via the Stereo Vision system, which are shown in Fig. 5.3

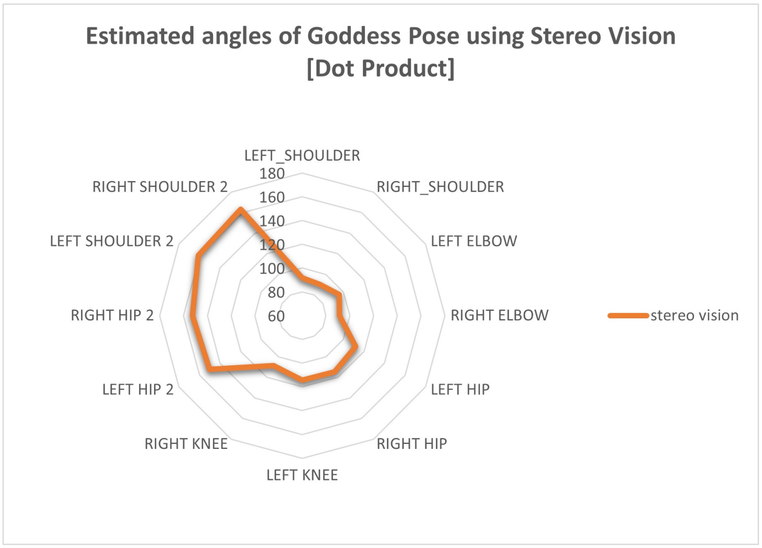


Figure 5.3: Stereo Vision system results of Goddess Pose, using Dot Product

Lastly, the results obtained using a single camera to compute angles via the Dot Product is shown in Fig. 5.4

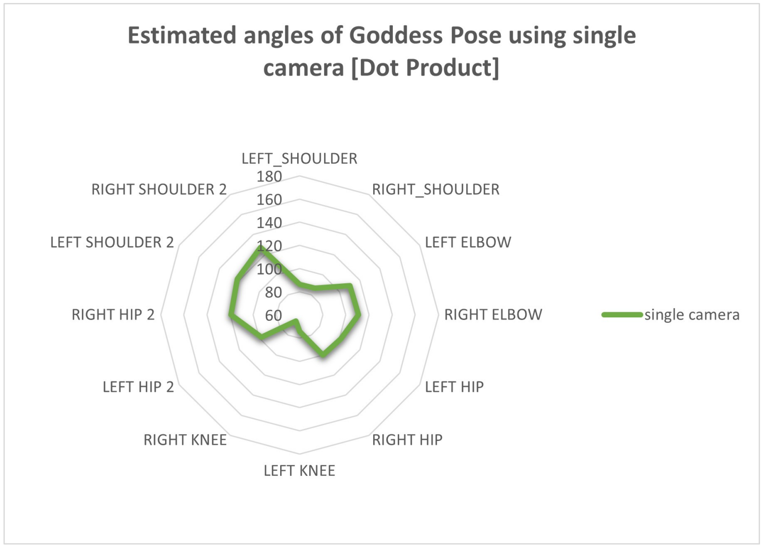


Figure 5.4: Single camera results of Goddess Pose, using Dot Product

Additionally, the results previously shown are integrated in Fig. 5.5 in order to evaluate the accuracy of both the Stereo Vision and Single camera systems and directly compare them with the results of the Kinect device.

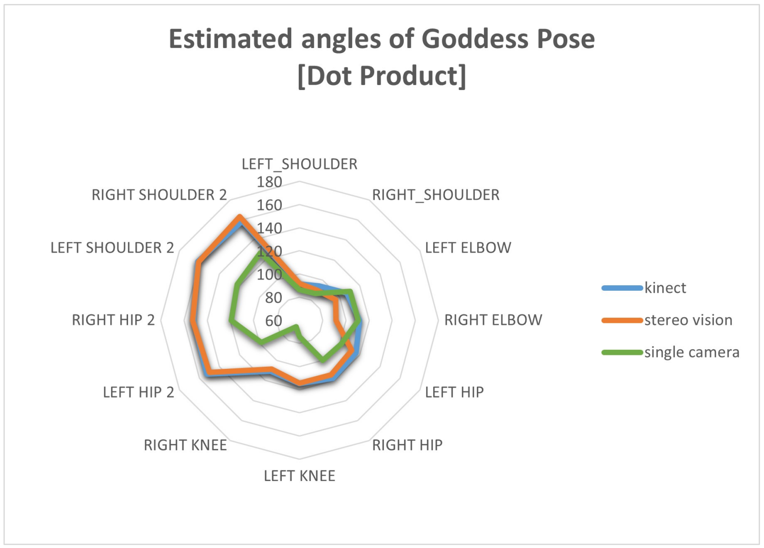


Figure 5.5: Results of the three computer vision systems, for the Goddess pose, using dot product

Lastly, based on the procedure used to obtain the values for the reference angles for each pose (Sec. 4.6.1), it is possible to plot them alongside the results obtained for each computer vision method. This can be seen in Fig. 5.6 which now incorporates the reference angles for the Goddess pose using the dot product method.

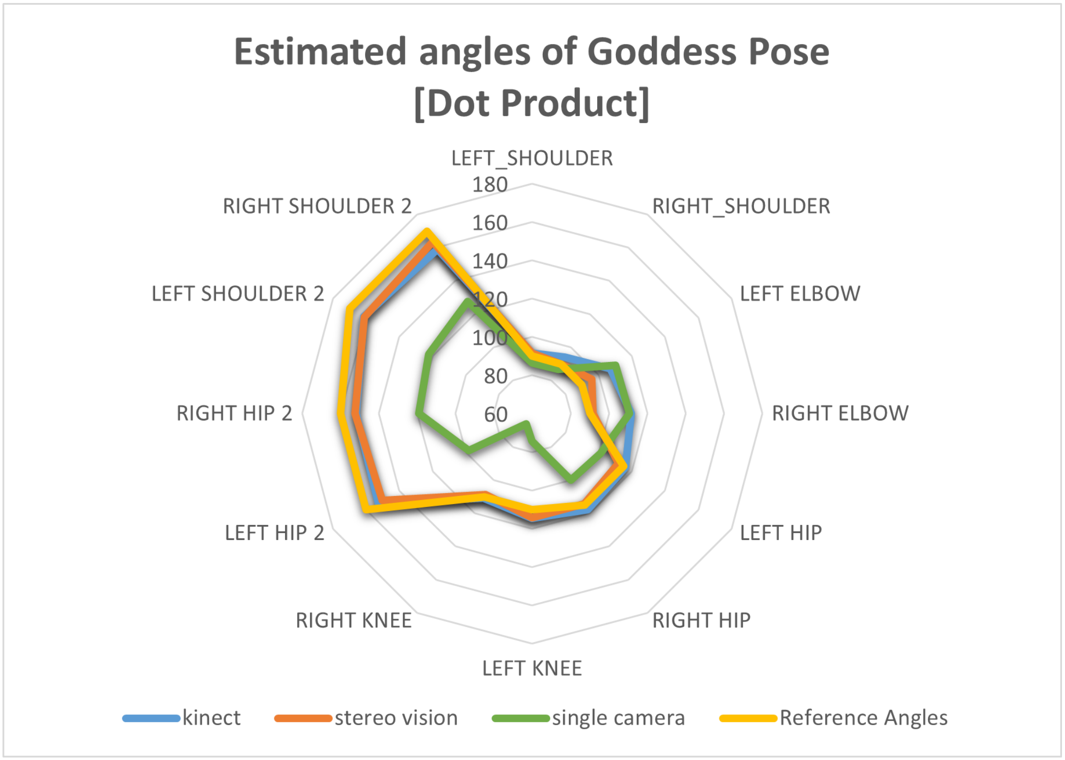


Figure 5.6: Results of the three computer vision systems, for the Goddess pose, alongside the obtained reference angles, using dot product

In Fig. 5.6, taking the reference angles plot as a starting point, we can clearly appreciate the similarity between it and the kinect and stereo vision results. This is not the case, however, for the single camera system, in which the keypoints coordinates information (x, y and z) provided by MediaPipe Pose alone were used to estimate joint angles.

5.2.2 Warrior II Pose

Another test example is shown in Fig. 5.7 where another volunteer is performing the Warrior II Pose.

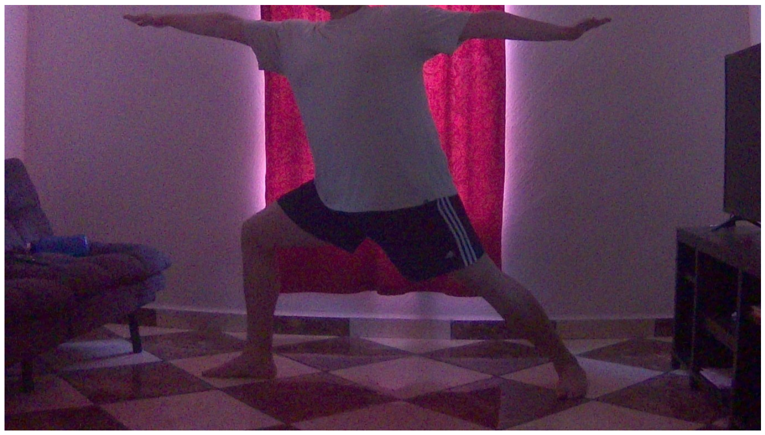


Figure 5.7: Second volunteer performing the Warrior II Pose

As it was done for the Goddess Pose, it was possible to generate a radar chart, shown in Fig. 5.8, integrating the results obtained using the three computer vision systems. This also allowed to visually evaluate the performance of the Stereo Vision and Single Camera systems, whilst taking the Kinect as reference. The plot also incorporates the reference angles obtained for the Warrior II pose.

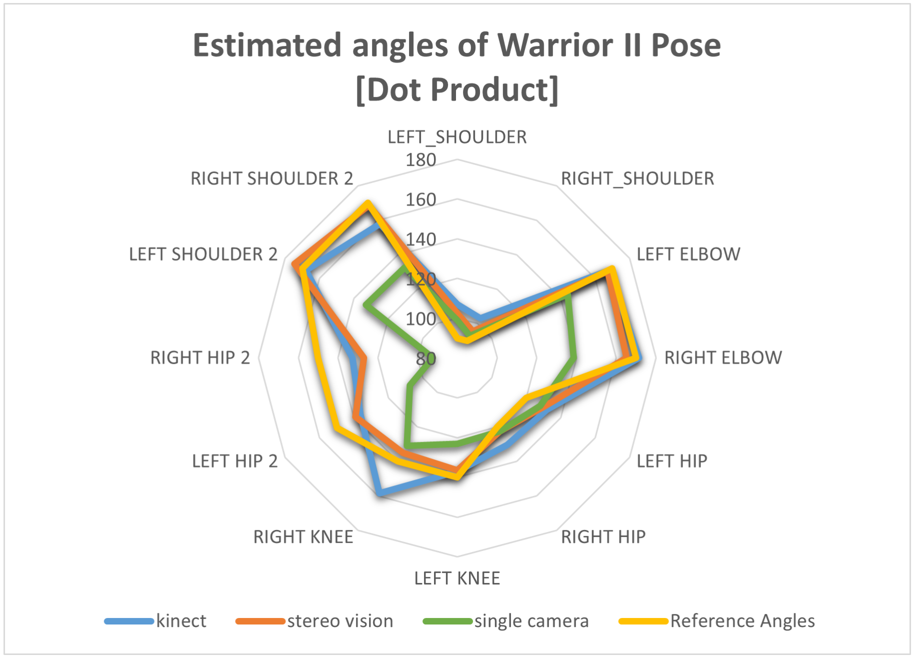


Figure 5.8: Results of the three computer vision systems, alongside the reference angles, for the Warrior II Pose, using dot product

Note that, for this particular pose and contrary to the results of the Goddess pose, there is a disparity between the plot of the reference angles and the rest of the systems (Kinect, Stereo Vision and Single camera). This could be due to a number of factors, for instance, because the rest of the systems failed to correctly estimate the pose and joint angles of the users. Also, because the reference angles were incorrectly calculated or, lastly, because the users failed to correctly perform the pose. It is suggested that the latter is the case, due to the similarity between the plots of the kinect and stereo vision systems. Similarly, there is a greater difference between the results obtained via the single camera and the rest of the systems.

5.2.3 Tree Pose

Lastly, the results obtained for the Tree Pose are presented as another volunteer performs said pose on Fig. 5.9

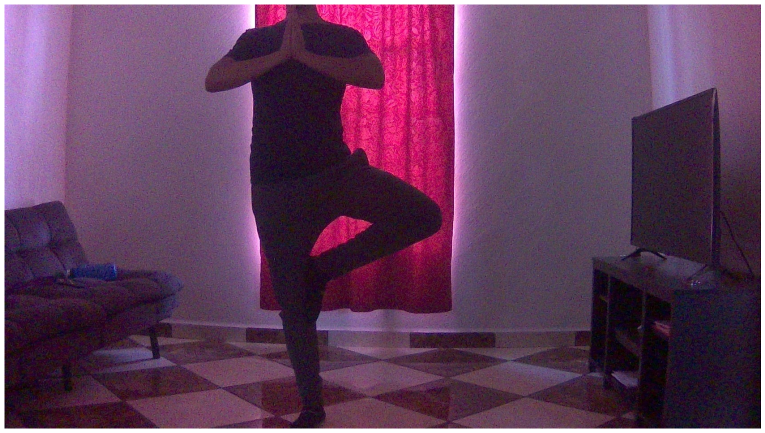


Figure 5.9: Third volunteer performing the Tree pose

Finally, the results of estimated angles, via the Dot Product, for the Tree Pose, including the obtained reference angles, are condensed in Fig. 5.10.

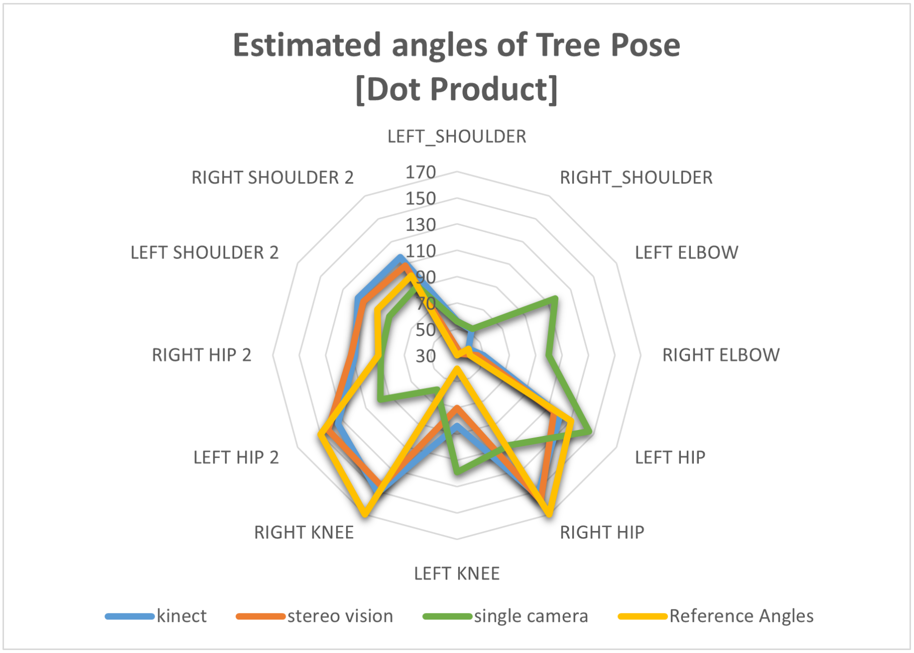


Figure 5.10: Results of the three computer vision systems, for the Tree pose, using Dot product

A similar situation to the previous pose happens as there is a disparity between the reference angles and the three computer vision systems used, however, as it was explained for the previous pose, this could be due to the users in general not fully correctly performing the pose. As before, the results obtained using the single camera system deviate greatly from the rest and, in particular, the reference angles.

5.3 Inverse Kinematics

The current section presents the results of the estimated angles of all three poses using the proposed Inverse Kinematics methodology. As it was done in the previous section, initially, the results for the Goddess pose are shown, followed by the Warrior II pose, and finally, the Tree pose. Results for all computer vision systems are presented.

5.3.1 Goddess Pose

Fig. 5.11 joins the results obtained by the three computer vision systems, for the Goddess pose, including the obtained reference angles for this methodology.

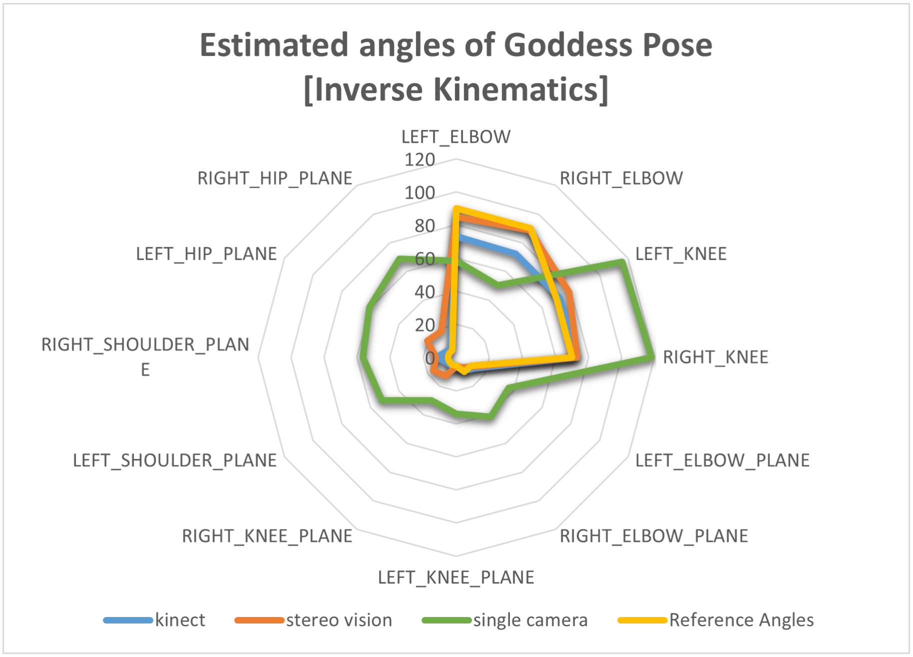


Figure 5.11: Results of the three computer vision systems, for the Goddess pose, using Inverse Kinematics

5.3.2 Warrior II Pose

Similarly, Fig. 5.12 presents the results of the computed angles for the Warrior II pose. Like before, a disparity is noted between the plot of the reference angles and the computer vision systems.

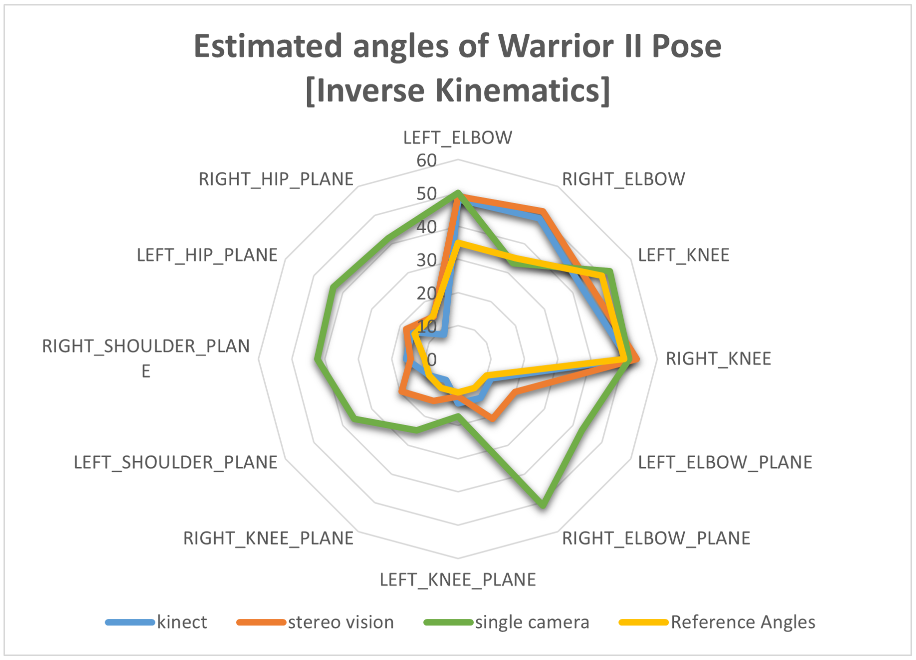


Figure 5.12: Results of the three computer vision systems, for the Warrior II pose, using Inverse Kinematics

5.3.3 Tree Pose

Lastly, the results for the Tree pose are depicted in Fig. 5.13, including the plot for the reference angles.

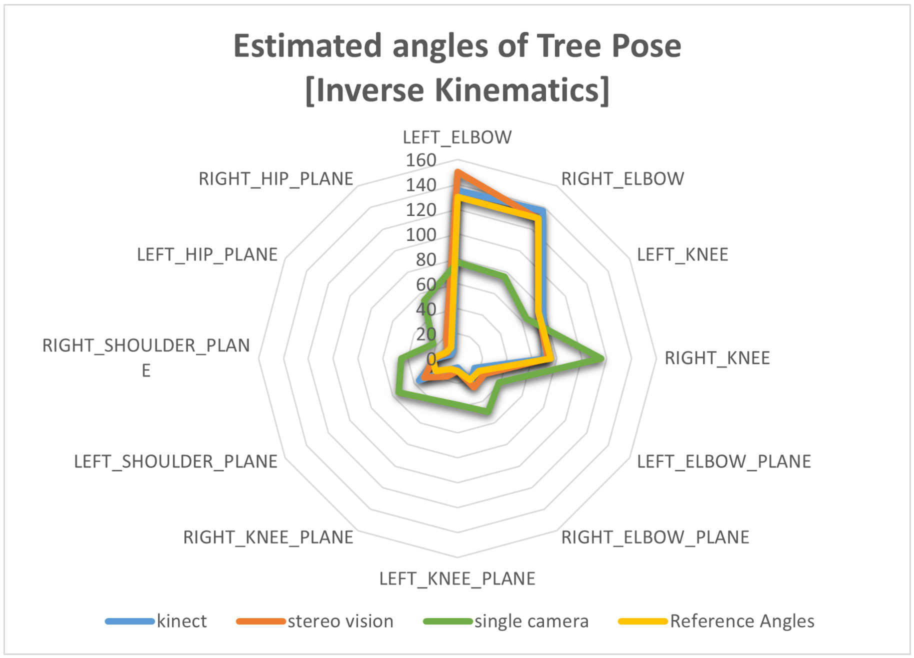


Figure 5.13: Results of the three computer vision systems, for the Tree pose, using Inverse Kinematics

Figures 5.11, 5.12 and 5.13 allow to appreciate the similarity of the results that were obtained via the kinect and stereo vision systems when compared with the reference angles for each pose. As it was the case for the Dot Product methodology, in general, the results obtained using the single camera system deviate notably from the rest and, in particular, from the reference angles.

5.4 Numerical Comparison of Angle Estimation Results

A summary is presented in Table 5.1 outlining the results obtained with each computer vision system and angle estimation method. The table expresses the results in terms of the root mean square error (RMSE) with respect to the Kinect. The RMSE is a useful evaluation metric that indicates the measure of differences between predicted or estimated values by a model and actual observed values [69]. The formula for the RMSE is shown in Eq. 5.1, where N indicates the amount of observations. By applying the square root to the quotient of the summation of squared differences and the amount of observed values, the RMSE expresses its results in the same units as the observed phenomenon.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(predicted_i - actual_i)^2}{N}} \tag{5.1}$$

The procedure to obtain the RMSE values consisted of several steps. Initially, the angle results for each Yoga pose, computer vision system and angle estimation method were averaged. For instance, the angle results that were obtained when all volunteers performed the Goddess Pose and when the Stereo Vision System and Dot Product method were used, were averaged. The same was done for the remaining poses, computer vision systems and angle estimation methods. For both Stereo Vision and Single Camera, the angle results for a particular pose and angle estimation method were compared with those obtained by the Kinect device by making use of Eq. 5.1. In this case, the N variable stands for the amount of angles that were compared, which is 12, regardless of the estimation method. Once this procedure was completed, the obtained RMSE values were averaged for every computer vision system and angle estimation method. For instance, three RMSE values were obtained for the Stereo Vision system and Dot product methods, as there were three yoga poses that were evaluated. These three RMSE values were averaged and the obtained value was placed in the corresponding cell in Table 5.1.

	Single camera	Stereo Vision
Dot product	32.88°	8.84°
Inverse Kinematics	32.84°	7.06°

Table 5.1: RSME for each computer vision system and angle estimation method with respect to the Kinect

The values shown in Table 5.1 indicate the average difference that each computer vision system and angle estimation method yield with respect to the Kinect. For example, on average, when using the Dot Product method, the Stereo Vision System estimates angles with an absolute error of 8.84° with respect to the Kinect.

5.5 Graphical User Interface

A graphical interface was designed to allow users to interact with the system. The GUI was designed and programmed using the Tkinter package included in the Tk Python package. It is an easy and friendly library for the development of graphical interfaces [70]. As seen in Fig. 5.14, the UI allows users to select the exercise, amount of time, and the difficulty level, such as "Easy", "Medium" and "Hard". The UI also incorporates a "Start" button, as well as an "End" button, which can be triggered at any point during an exercise, before running time ends.

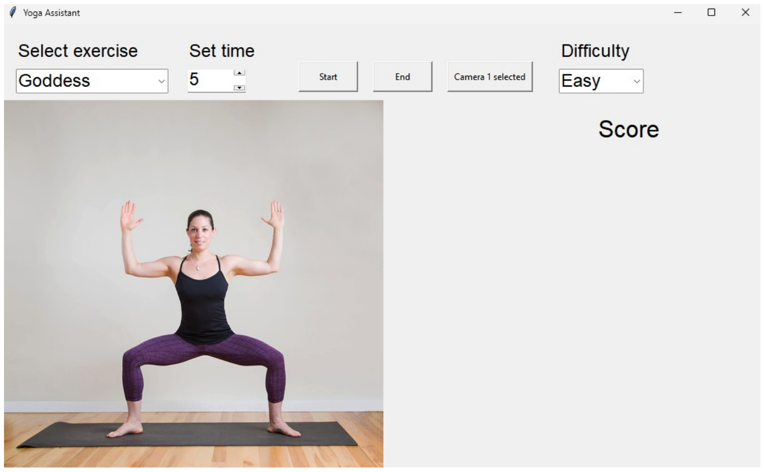


Figure 5.14: Graphical User Interface

At the right of the UI, users are provided with their score, updated in real time. Additionally, they are able to see the stick figure and color scale system, such as the example provided in Fig. 4.12, providing feedback regarding which joints should be corrected. Prior to the beginning of the exercise, the interface displays a reference image indicating how the user should perform the pose they've selected.

Once the exercise begins, the reference image is replaced by the live video feed. On top of the feed, the user is able to visualize a countdown displaying the remaining time, which is shown on the top-right corner of the image.

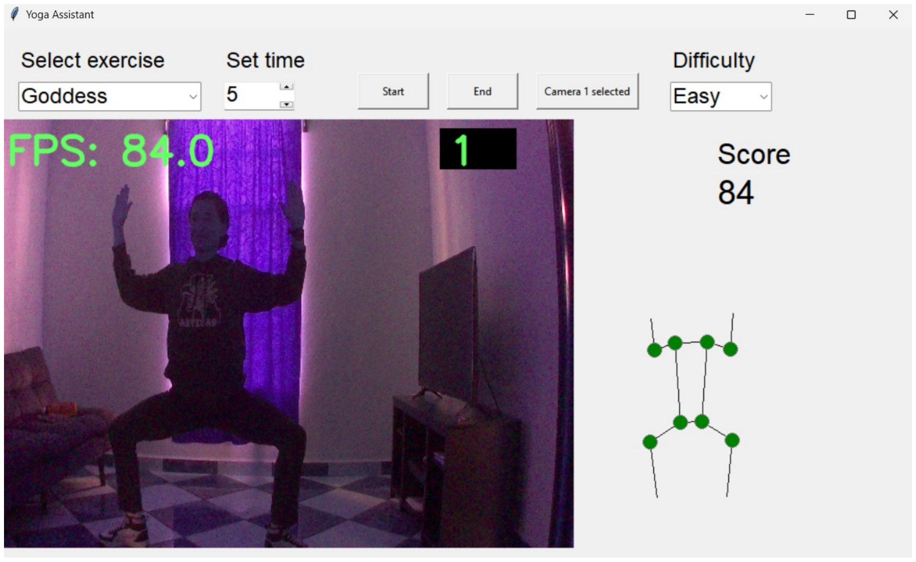


Figure 5.15: Graphical User Interface while a volunteer is performing the Goddess pose

Additionally, once the exercise ends, a window pops up, displaying a radar chart. The radar chart plots the reference angles of the exercise, as well as the averaged angles the user obtained during the exercise. The aim of the radar chart is to provide the user with additional feedback, such that they can learn overall which joints could be improved and by how much.

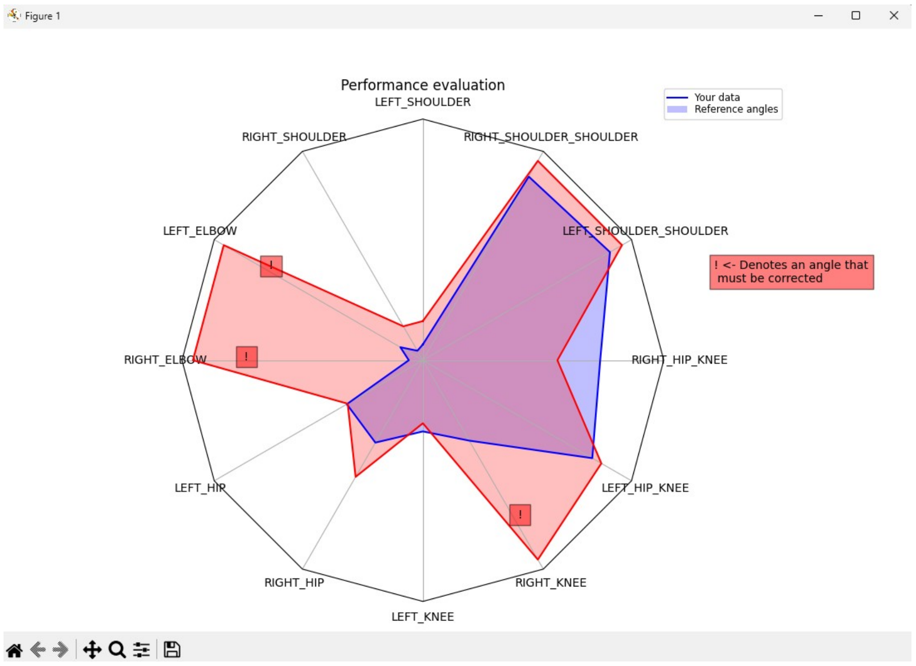


Figure 5.16: Example of the performance plot

The chart displays exclamation marks, highlighted in red, for each joint angle that could be improved. An exclamation is displayed if the percent error between the reference value and the user's value exceeds a certain threshold. The specific value of the threshold depends on the difficulty level that the user selected. If the user chose the "Easy" level, the threshold is 35%, for the "Medium" level, 25%, and "Hard", 15%.

5.6 Full System Hosted on the Jetson Nano

Lastly, the complete system was assembled as it is shown in Fig. 5.17. The two IMX219 cameras are connected to the Jetson Nano. The Jetson Nano lays inside the black case, which was acquired from Waveshare. For ease of use, the system incorporates a 7-inch touch screen. The touch screen allows user to interact without the need for a mouse and keyboard.

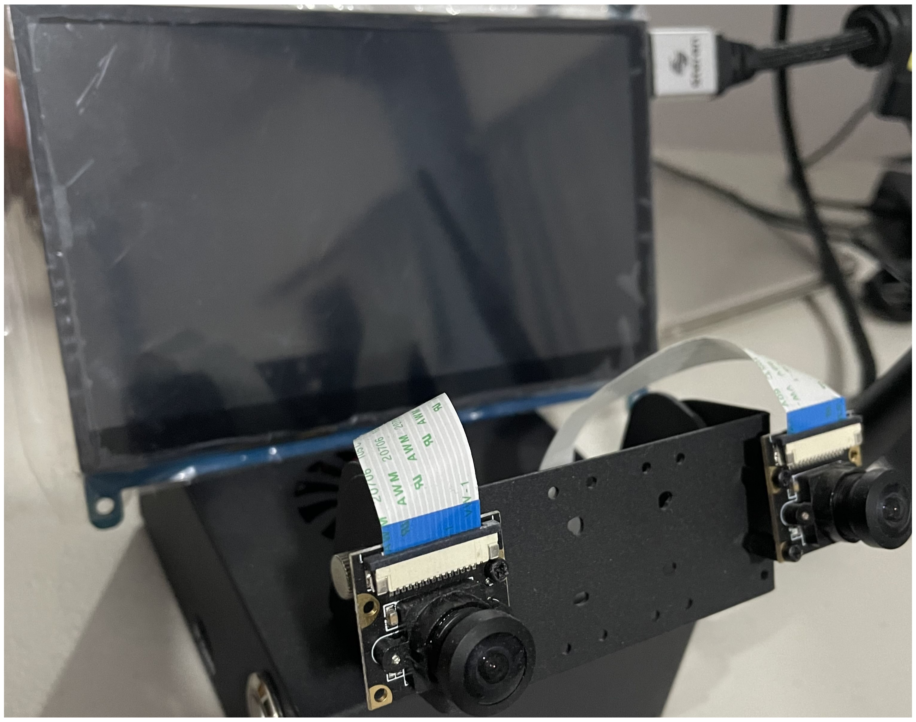


Figure 5.17: Full system hosted on the Jetson Nano

The Python program's performance was analyzed by keeping track of the video feed resolution in frames per second (fps). It was observed that the Jetson Nano was able to achieve an average resolution of 50 fps while displaying feed from either of the IMX219 cameras. The Jetson Nano's performance was monitored using libraries for monitoring and control of several statistics of the Jetson, such as CPU, GPU, and Memory. In particular, the *jetson-stats* tool allowed to monitor the Jetson Nano while hosting both computer vision system, as well as performing both angle estimation methods. It was observed that memory consumption peaked when using Stereo Vision, regardless of the angle estimation method, as it maintained an average memory usage of >400 MB's. Similarly, memory consumption was observed to be lower when using a single camera, as it remained at around 350-400 MB's.

Chapter 6

Discussion and future work

In this work, a system for real-time evaluation of Yoga poses was developed. This work involved several stages, such as enabling MediaPipe Pose for pose estimation, acquiring and implementing the hardware necessary for Computer Vision, defining angle estimation methods, and designing a protocol to evaluate user pose in order to provide feedback. Lastly, the full system was embedded on a portable but powerful device, such as the Jetson Nano, which also allowed user interaction in real time via a touchscreen and graphical interface.

Two systems for computer vision were implemented in order to evaluate and compare their respective performances for pose detection and angle estimation, such as a single camera and a stereo vision system. Similarly, two methods for angle estimation were implemented. On one hand, this work implemented the Vector Dot Product, as it can be used to obtain the angle between two N-dimensional vectors, such as the 3D vectors defined in this work. On the other hand, Inverse Kinematics is popular in the context of Robotics to determine the necessary angle configuration of a kinematic chain. In this case, the Inverse Kinematics-based procedure made use of a geometric approach to obtain the angle between several joints of the body. One of the main advantages of implementing an approach based on geometry is its computational efficiency and simplicity.

With the aid of the Kinect device, it was possible to verify that the Stereo Vision System is capable of accurate estimation of several 3D joint angles of the human body, using both angle estimation methods described. The Kinect device allowed to corroborate that Stereo Vision is more suitable for depth and angle estimation of real-world objects, instead of single camera methods such as the depth estimations provided by MediaPipe Pose. This can be seen in Figs. 5.6, 5.8, 5.10, 5.11, 5.12, and 5.13, as well as Table 5.1, in which the results obtained via the Kinect device were taken as the reference and, thus, represent the desired values to be obtained by the Stereo Vision and single camera systems. As it can be seen, the Stereo Vision System managed to obtain more similar results to those of the Kinect. Results prove the Stereo Vision System is capable of accurate 3D pose detection, and, therefore, reliable for providing the correct feedback and perform precise pose evaluation.

This work represents great relevance in the field of pose estimation and pose evaluation. As seen in the results, the combination of 2D pose detection with depth estimation via Stereo Vision allowed to perform accurate 3D human pose estimation. This is highly relevant in the context of pose evaluation, as the pose detection system needs to be able to accurately estimate human pose in order to provide correct feedback. Several works in the literature have implemented pose evaluation systems by making use of pose estimation frameworks such as OpenPose, PoseNet and MoveNet. However, these systems only take into consideration 2D information, which can result in inaccurate pose estimation and, therefore, evaluation [15], [16], [17], [29], [63], [64], [65]. Performing pose estimation and evaluation using 2D information only can limit the use cases of the system, as it may become inaccurate when the user is not facing front towards the camera [66], however, taking into account 3D information represents a more robust system and, ideally, makes it insensible to user orientation with respect to the camera, which is the case for this work. Similarly, while other works have made use of MediaPipe Pose, they either consider only x and y keypoints coordinates or rely on the depth estimations that it provides [28], [61], [62]. The implemented Stereo Vision System in this work allows for improved pose estimation and evaluation, as it performs accurate depth estimation, which represents a more reliable system that is correctly capturing the shape and pose of the person. The above is condensed in Table 6.1, which outlines several works that implement similar techniques to evaluate yoga exercises, along with their main advantages and limitations, as well as this work, thus highlighting the relevance of this thesis project within the state of the art.

Year, [ref]	Pose estimation framework	Evaluation method	Feedback form (Platform)	Advantages	Limitations
2019 [15]	OpenPose, Single webcam	2D Joint angles	Red-green color scale, "Perfect" to "Bad" scale	Intuitive feedback	Limited to 2D pixel coordinates
2021 [16]	OpenPose, Single webcam	2D Joint angles	Step by step instructions (4 GB RAM machine)	Guides users step by step	Limited to 2D pixel coordinates
2024 [29]	PoseNet, Single webcam	2D Joint angles	Text, audio messages to correct posture (Web App)	Specific and detailed audio-based feedback	Internet connection required, limited to 2D coordinates
2023 [61]	MediaPipe Pose Single webcam	2D Joint angles	Success/negative message (64-bit OS)	Requires a lightweight platform	Feedback is not joint-specific
2020 [63]	PoseNet, Single webcam	2D Joint angles	Correct/Wrong message, more/less/ok message for each joint	Feedback is joint-specific	Difficult to read feedback for each joint
2024 [66]	MoveNet, Single webcam	2D Joint angles	Green/white skeleton drawing for correct/incorrect pose (Web App)	Simple yet intuitive feedback	Requires the user to face front towards the camera
2024, [This work]	MediaPipe Pose, Single camera/ Stereo Vision	*3D* Joint angles	Red to green color scale, live 0-100 score (NVIDIA Jetson Nano)	Joint-specific intuitive feedback, estimation accuracy validated by kinect	Does not guide user step by step

Table 6.1: Comparison between this work and the state of the art

A Graphical User Interface was designed for users to interact with the system. The GUI allowed to configure several exercise parameters and provided the user with real-time feedback to improve performance. Via the GUI, users can select the desired exercise, difficulty level and execution time. The GUI provides real-time feedback in the form of a 0-100 score, as well as a color-scale system to visually indicate improvement opportunities. Additionally, after finishing an exercise, the GUI displays a radar chart with the average angle values and indicates which should the user pay more attention to in order to improve.

Lastly, in terms of pose evaluation, this work proposed a protocol to assess the performance of yoga poses in real time. The developed system and evaluation methodology will be of aid to users when performing Yoga, as it has proven to be capable of accurate pose detection, and, therefore, correct pose evaluation. The system proved to be able to provide correct feedback in real time without the need of a certified instructor. The implementation of a visual color scale and point-scoring system allows for intuitive feedback that is clear for the user to understand. This will reduce injury risk and instead help improve mental and physical health of individuals.

Future work

Future work should focus on adding more exercises for users to choose from. Additionally, the pose evaluation protocol designed in this work could be applied to further disciplines such as weight training and physiotherapy. Moreover, while this work focused on static exercises, in which a person is only required to hold a pose for a certain amount of time, it can be extended to exercises in which body pose varies over time. In particular, the Dynamic Time Warping (DTW) algorithm, as well as the Long Short-Term Memory (LSTM) network could prove useful to enable evaluation of dynamic exercises. The DTW algorithm provides a method to compare two time series similar in shape and behaviour but at differing times [71]. On the other hand, LSTMs are capable of remembering time-dependent information, making them suitable for handling and predicting data sequences that vary over time [72].

Bibliography

[1] NVIDIA Developer. Get started with jetson nano developer kit. `https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit`. Accessed: 2024-03-02.

[2] Google AI for Developers. Pose landmark detection guide. `https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker`. Accessed: 2024-03-02.

[3] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204*, 2020.

[4] Abhijit Jana. *Kinect for Windows SDK Programming Guide*. 2012.

[5] NI Vision Concepts Help. Parts of a stereo vision system. `https://www.ni.com/docs/en-US/bundle/ni-vision-concepts-help/page/stereo_parts_of_a_stereo_vision_system.html`. Accessed: 2024-04-10.

[6] Mathworks. What is inverse kinematics? `https://la.mathworks.com/discovery/inverse-kinematics.html`. Accessed: 2024-06-19.

[7] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020.

[8] Sam Cubero. *Industrial Robotics: Theory, Modelling and Control*. 2012.

[9] Luxmi Sharma. Benefits of yoga in sports –a study. *International Journal of Physical Education, Sports and Health IJPESH*, 1, 2015.

[10] Gyanesh Kumar Tiwari. Yoga and mental health: An underexplored relationship. volume 4, 2016.

[11] Christine Wiese, David Keil, Anne S. Rasmussen, and Rikke Olesen. Injury in yoga asana practice: Assessment of the risks. *Journal of Bodywork and Movement Therapies*, 23, 2019.

[12] Faisal Bin Ashraf, Muhammad Usama Islam, Md Rayhan Kabir, and Jasim Uddin. Yonet: A neural network for yoga pose classification. *SN Computer Science*, 4, 2023.

[13] Utkarsh Bahukhandi and Shikha Gupta. Yoga pose detection and classification using machine learning techniques. *Int Res J Mod Eng Technol Sci*, 3(12):13–15, 2021.

[14] Laia Turmo Vidal, Elena Márquez Segura, Luis Parrilla Bel, and Annika Waern. Training technology probes across fitness practices: Yoga, circus and weightlifting. Association for Computing Machinery, 4 2020.

[15] Maybel Chan Thar, Khine Zar Ne Winn, and Nobuo Funabiki. A proposal of yoga pose assessment method using pose detection for self-learning. 2019.

[16] Ayush Gupta and Ashok Jangid. Yoga pose detection and validation. 2021.

[17] Fazil Rishan, Binali De Silva, Sasmini Alawathugoda, Shakeel Nijabdeen, Lakmal Rupasinghe, and Chethana Liyanapathirana. Infinity yoga tutor: Yoga posture detection and correction system. 2020.

[18] Edwin W. Trejo and Peijiang Yuan. Recognition of yoga poses through an interactive system with kinect device. 2018.

[19] Chhaihuoy Long, Eunhye Jo, and Yunyoung Nam. Development of a yoga posture coaching system using an interactive display based on transfer learning. *Journal of Supercomputing*, 78, 2022.

[20] Yash Agrawal, Yash Shah, and Abhishek Sharma. Implementation of machine learning technique for identification of yoga poses. In *2020 IEEE 9th international conference on communication systems and network technologies (CSNT)*, pages 40–43. Ieee, 2020.

[21] Shrajal Jain, Aditya Rustagi, Sumeet Saurav, Ravi Saini, and Sanjay Singh. Three-dimensional cnn-inspired deep learning architecture for yoga pose recognition in the real-world environment. *Neural Computing and Applications*, 33:6427–6441, 2021.

[22] Josvin Jose and S Shailesh. Yoga asana identification: a deep learning approach. In *IOP Conference Series: Materials Science and Engineering*, volume 1110, page 012002. IOP Publishing, 2021.

[23] Shakti Kinger, Abhishek Desai, Sarvarth Patil, Hrishikesh Sinalkar, and Nachiket Deore. Deep learning based yoga pose classification. In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, volume 1, pages 682–691. IEEE, 2022.

[24] Santosh Kumar Yadav, Amitojdeep Singh, Abhishek Gupta, and Jagdish Lal Raheja. Real-time yoga recognition using deep learning. *Neural computing and applications*, 31:9349–9361, 2019.

[25] V Rathikarani, S Abarna, and K Vijayakumar. Classification of yoga pose using pretrained convolutional neural networks. *Journal of Pharmaceutical Negative Results*, pages 3798–3805, 2022.

[26] Silky Goel, Shlok Mohanty, and Snigdha Markanday. Classification of yoga pose using pre-trained cnn models and machine learning classifiers. In *2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)*, pages 1–6. IEEE, 2022.

[27] Shubham Garg, Aman Saxena, and Richa Gupta. Yoga pose classification: a cnn and mediapipe inspired deep learning approach for real-world application. *Journal of Ambient Intelligence and Humanized Computing*, 14(12):16551–16562, 2023.

[28] Vedangi Agarwal, Konark Sharma, and Abha Kiran Rajpoot. Ai based yoga trainer-simplifying home yoga using mediapipe and video streaming. In *2022 3rd International Conference for Emerging Technology (INCET)*, pages 1–5. IEEE, 2022.

[29] Aditya Potdar, Jay Bhanushali, Sarvagya Singh, and Kanchan Dabre. Yoga pose classification and correction using posenet. In *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, volume 2, pages 1–6. IEEE, 2024.

[30] Cheng-Hsien Lin, Shih-Wei Shen, Irin Tri Anggraini, Nobuo Funabiki, and Chih-Peng Fan. An openpose-based exercise and performance learning assistant design for self-practice yoga. In *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, pages 456–457. IEEE, 2021.

[31] NVIDIA. Nvidia jetson nano. `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/`. Accessed: 2024-03-02.

[32] NVIDIA. Bringing the power of modern ai to millions of devices, 2022.

[33] Arunava Nag and Sanket Deshmukh. Real time tracking system using 3d vision. 2015.

[34] Google AI for Developers. Mediapipe solutions guide. `https://ai.google.dev/edge/mediapipe/solutions/guide`. Accessed: 2024-03-02.

[35] Google AI for Developers. Vision examples. `https://ai.google.dev/edge/mediapipe/solutions/examples`. Accessed: 2024-03-02.

[36] Valentin Bazarevsky and Ivan Grishchenko. On-device, real-time body pose tracking with mediapipe blazepose. https://research.google/blog/on-device-real-time-body-pose-tracking-with-mediapipe-blazepose/?source=post_page-----d8689d06b7c4--------------------------------. Accessed: 2024-03-02.

[37] Valentin Bazarevsky and Ivan Grishchenko. Google ai blog: On-device, real-time body pose tracking with mediapipe blazepose, 2020.

[38] Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, Andrew Blake, Mat Cook, Alex Kipman, Mark Finocchio, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56, 2013.

[39] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.

[40] John Winn and Jamie Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 37–44. IEEE, 2006.

[41] NI Vision Concepts Help. Stereo vision. https://www.ni.com/docs/en-US/bundle/ni-vision-concepts-help/page/stereo_vision.html. Accessed: 2024-04-10.

[42] Rahul Kala. *Basics of Autonomous Vehicles.* 2016.

[43] Habib Izadkhah. Chapter 2 - a review of machine learning. In Habib Izadkhah, editor, *Deep Learning in Bioinformatics*, pages 9–30. Academic Press, 2022.

[44] Vincenzo Pesce, Andrea Colagrossi, and Stefano Silvestrini. *Modern Spacecraft Guidance, Navigation, and Control: From System Modeling to AI and Innovative Applications.* 2022.

[45] Serdar Kucuk and Zafer Bingul. Robot kinematics: Forward and inverse kinematics. In Sam Cubero, editor, *Industrial Robotics*, chapter 4. IntechOpen, Rijeka, 2006.

[46] Marko B. Popovic and Matthew P. Bowers. 2 - kinematics and dynamics. In Marko B. Popovic, editor, *Biomechatronics*, pages 11–43. Academic Press, 2019.

[47] Nacer Hadidi, Mohamed Bouaziz, Chawki Mahfoudi, and Mohamed Zaharuddin. Geometric approach to solving inverse kinematics of six dof robot with spherical joints. 2023.

[48] Annisa Jamali, Raisuddin Khan, and Md Mozasser Rahman. A new geometrical approach to solve inverse kinematics of hyper redundant robots with variable link length. In *2011 4th International Conference on Mechatronics (ICOM)*, pages 1–5. IEEE, 2011.

[49] David DeMers and Kenneth Kreutz-Delgado. 4 - inverse kinematics of dextrous manipulators. In Omid Omidvar and Patrick van der Smagt, editors, *Neural Systems for Robotics*, pages 75–116. Academic Press, Boston, 1997.

[50] Rick Parent. *Computer animation: Algorithms and techniques.* 2012.

[51] CMU Perceptual Computing Lab. Openpose. `https://github.com/CMU-Perceptual-Computing-Lab/openpose`. Accessed: 2024-06-25.

[52] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[53] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286, 2018.

[54] Community TensorFlow.js. Next-generation pose detection with movenet and tensorflow.js. `https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html`. Accessed: 2024-06-25.

[55] COCO dataset. Coco - common objects in context. `https://cocodataset.org/#home`. Accessed: 2024-04-02.

[56] Max Planck Institut Informatik. Mpii human pose database. `http://human-pose.mpi-inf.mpg.de/`. Accessed: 2024-04-02.

[57] COCO dataset. Coco - common objects in context. `https://cocodataset.org/#detection-eval`. Accessed: 2024-04-02.

[58] Google AI for Developers. Mediapipe blazepose ghum 3d. `https://developers.google.com/ml-kit/images/vision/pose-detection/pose_model_card.pdf`. Accessed: 2024-02-02.

[59] TensorFlow. Pose estimation. `https://www.tensorflow.org/lite/examples/pose_estimation/overview#:~:text=Pose%20estimation%20refers%20to%20computer,shows%20up%20in%20an%20image`. Accessed: 2024-02-02.

[60] Google APIs. Movenet.singlepose model card. `https://storage.googleapis.com/movenet/MoveNet.SinglePose%20Model%20Card.pdf`. Accessed: 2024-04-02.

[61] Isha Chaudhary, Nongmeikapam Thoiba Singh, Mahak Chaudhary, and Komal Yadav. Real-time yoga pose detection using opencv and mediapipe. In *2023 4th International Conference for Emerging Technology (INCET)*, pages 1–5. IEEE, 2023.

[62] T Anuradha, N Krishnamoorthy, CS Pavan Kumar, LV Narasimha Prasad, Anilkumar Chunduru, and Usha Moorthy. A method for specifying yoga poses based on deep learning, utilizing opencv and media pipe technologies. *Scalable Computing: Practice and Experience*, 25(2):739–750, 2024.

[63] Sonali Muley, Prateek Mahajan, Pratik Medidar, Harish Chavan, and Prashant Patil. Yoga guidance using human pose estimation. *IRJMETS*, 2(9):1533–1537, 2020.

[64] Girija Gireesh Chiddarwar, Abhishek Ranjane, Mugdha Chindhe, Rachana Deodhar, and Palash Gangamwar. Ai-based yoga pose estimation for android application. *Int J Inn Scien Res Tech*, 5(2020):1070–1073, 2020.

[65] Aman Upadhyay, Niha Kamal Basha, and Balasundaram Ananthakrishnan. Deep learning-based yoga posture recognition using the y_pn-mssd model for yoga practitioners. In *Healthcare*, volume 11, page 609. MDPI, 2023.

[66] Amey Parle, Rugved Shinde, Rahul Chougule, and Shruti Agrawal. Yogawise: Enhancing yoga with intelligent real time tracking using tensorflow movenet. In *2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC-ROBINS)*, pages 498–505. IEEE, 2024.

[67] Sony. Imx219. `https://developer.sony.com/vision-sensing-processor/hardware-overview/sensors/cmos/imx219`. Accessed: 2024-03-02.

[68] Waveshare Electronics. Imx219 camera series, 8mp, applicable for jetson nano and raspberry pi, options for fov and night vision function. `https://www.waveshare.com/imx219-160-camera.htm`. Accessed: 2024-03-02.

[69] Deepak Kumar Sharma, Mayukh Chatterjee, Gurmehak Kaur, and Suchitra Vavilala. 3 - deep learning applications for disease diagnosis. In Deepak Gupta, Utku Kose, Ashish Khanna, and Valentina Emilia Balas, editors, *Deep Learning for Medical Applications with Unique Data*, pages 31–51. Academic Press, 2022.

[70] Python documentation. Graphical user interfaces with tk. `https://docs.python.org/3/library/tk.html`. Accessed: 2024-06-02.

[71] Hailin Li. Time works well: Dynamic time warping based on time weighting for time series data mining. *Information Sciences*, 547:592–608, 2021.

[72] Khalid K. Al-jabery, Tayo Obafemi-Ajayi, Gayla R. Olbricht, and Donald C. Wunsch II. 4 - selected approaches to supervised learning. In Khalid K. Al-jabery, Tayo Obafemi-Ajayi, Gayla R. Olbricht, and Donald C. Wunsch II, editors, *Computational Learning Approaches to Data Analytics in Biomedical Applications*, pages 101–123. Academic Press, 2020.

Stereo Vision System based on the NVIDIA Jetson Nano for Real-time Evaluation of Yoga Poses

Eric Williams-Linera
Electronics Department, INAOE
Puebla, Mexico
eric.williams@inaoep.mx

Juan Manuel Ramírez-Cortés
Electronics Department, INAOE
Puebla, México
jmram@inaoep.mx

***Abstract*—Yoga has become increasingly popular in recent years as it offers several benefits through physical, mental, and spiritual practices. Although online resources offer the possibility of self-training at home without the need for an instructor, unsupervised training can result in injury, as users are not provided suggestions on how to improve. In this work, a Stereo Vision System, hosted on an NVIDIA Jetson Nano, was developed to evaluate Yoga poses in real time. The MediaPipe Pose framework and Stereo Vision System were enabled to perform 3D angle estimation of several joints of the human body. These angles were used to design and implement a methodology to evaluate user performance and provide them with feedback. The proposed method provides a 0-100 score, as well as a color scale system to visually indicate how a user can improve as they are performing a pose. Results prove the Stereo Vision System can perform accurate 3D Pose Detection and Angle Estimation, and is, therefore, reliable for providing accurate and correct feedback in real time. The proposed system will aid in minimizing injury risk while improving physical and mental health of individuals that practice Yoga.**

***Keywords*—Yoga, MediaPipe, Stereo Vision, Jetson Nano, Pose Evaluation, Body Angles**

I. Introduction

Yoga is a popular and ancient discipline that combines physical, mental, and spiritual practices to provide harmony between body and mind. Additionally, it offers several benefits in the form of improved flexibility, muscle strength, cardiovascular health, as well as mental and psychological health [1], [2]. The growing amount of open access online resources, such as video tutorials, have allowed individuals to practice Yoga at home without the need for an instructor. However, practice of Yoga without aid and supervision can increase injury risk, as personalized feedback and improvement suggestions are not provided to users. Yoga-related injuries mainly include muscle sprain and strain, as well as general pain [3].

While many works have implemented systems and algorithms for classification tasks [4], recent works have shifted focus towards evaluation of Yoga poses. Several studies have made use of wearable devices to monitor user performance [5]. Other works have implemented computer vision systems for Yoga poses identification and assessment. Such systems include Kinect devices, as well as Webcams and even mobile applications [6], [7], [8], [9], [10]. Moreover, the addition of Machine Learning algorithms could offer new possibilities and allow for more robust and accurate systems [8].

In this work, a Stereo Vision System for the evaluation of three Yoga poses in real time was developed. The poses applied to this work consist in Goddess, Warrior II and Tree. The system, comprised of two IMX219 cameras, was hosted on an NVIDIA Jetson Nano, a small but powerful computer, suitable for running multiple neural networks in parallel. The Stereo Vision System allowed to implement a methodology to assess user performance of Yoga poses to reduce injury risk while improving health.

II. Related work

Several works have focused on the implementation of algorithms for the identification of Yoga poses, mainly through Convolutional Neural Networks, as well as learning algorithms such as Support Vector Machines and Decision Trees. Recent works have focused on the development of algorithms for the evaluation and correction of Yoga poses. In [6], a Yoga pose assessment methodology for self-learning was developed. Using OpenPose for keypoints detection, and a single PC camera, the proposed method calculates several 2D angles of a user's body in order to compare them with those of an instructor. The system classifies the Yoga pose into four levels depending on the average angle difference, which allows learners to understand the evaluation results.

In [7], a novel yoga pose validation system was developed. Using computer vision techniques, the system guides users through a series of steps necessary to correctly perform a particular Yoga exercise. Similarly, their work makes use of OpenPose to obtain x and y coordinates of several joints in order to calculate various angles and joint distances. The obtained angles and distances are compared with those of the current step the user is performing, once the user correctly performs the current step, they are allowed to continue on to the next one.

In [8], a Yoga coaching system, based on transfer learning, was developed. The system allows users to select from 14 different Yoga poses on an interactive graphical interface. A MobileNet model was trained on a custom-made dataset consisting of images collected from volunteers. The MobileNet model was able to achieve an 98.43% accuracy. Additionally, the system provides feedback in the form of a ”Correct” or ”Incorrect” message, as well as instructions to correct the pose. Feedback is based on several joint angles calculated using the MediaPipe Pose framework.

In [9], a Yoga posture detection and correction system was developed. The system is capable of detecting six different poses and provide feedback using a mobile application. Two pose detection models were tested, using OpenPose and a Mask RCNN model, respectively. Both models were used for the detection of keypoints, which were fitted to a prediction model for the identification of the Yoga posture the user is currently attempting to perform. The result is then streamed to the mobile application. It was found that the OpenPose model produced better results, however, it had difficulty differentiating between similar poses.

In [10], Trejo and Yuan presented an interactive system using the Kinect V2 device for the recognition and monitoring of Yoga poses. The system is able to track and recognize six Yoga poses and enables voice commands for the user to interact with the system, allowing them to correct their posture in real time. Using the Adaboost algorithm, a robust database was built for the detection of poses, which achieved an overall confidence value of 92%.

III. Proposed Yoga Pose Evaluation System

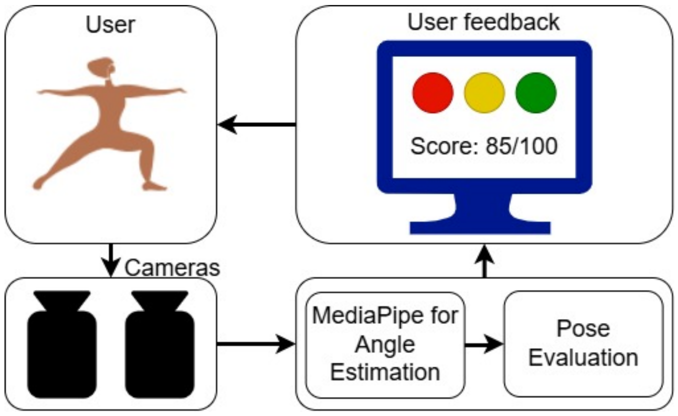


Fig. 1: Block diagram of the Yoga pose evaluation system

The proposed method, of which the block diagram can be seen in Fig. 1, makes use of a Stereo Vision System, powered by a Jetson Nano. The system takes as input two images of a user performing a particular Yoga pose. With the input images, MediaPipe Pose is used to perform pose estimation to obtain x and y image pixel coordinates for several body joints. Through the Stereo Vision System, depth estimation is performed to obtain the z coordinate for each detected keypoint. With the x, y and z coordinates of each joint, the system calculates several joint angles of the user's body. Once all necessary angles are estimated, they are compared with the ideal reference angles for the particular pose the user is performing. A point-scoring system was proposed to determine the accuracy and correctness with which the user performs the pose. The user is provided real-time feedback in the form of a 0-100 score. Additionally, a color scale, ranging from Green (Excellent) to Red (Very bad) was implemented, in which several colored circles are drawn on the video feed, to indicate which joints are being correctly performed, and which are not. Users are able to interact with a Graphical Interface (GUI) which allows them to select the desired Yoga pose and the amount of time they'd like to perform it. Additionally, a validation procedure was implemented to verify the Stereo Vision System's accuracy for pose detection and angle estimation. The validation procedure consisted in programming a Kinect V1 device to estimate the same angles as the Stereo Vision System, such that their respective results could be compared.

A. Stereo Vision System

A Stereo Vision system consists of two cameras oriented towards the same object, with the aim of estimating its depth. Fig. 2 depicts the Stereo Vision System implemented in this work. The goal of the Stereo System consists in identifying the same pixel on both camera frames, with the aim of obtaining a horizontal disparity which is used to estimate the depth of the object on interest [11]. For this particular work, the pixel in question consists of a particular keypoint (e.g. left wrist, right elbow), obtained via MediaPipe Pose.

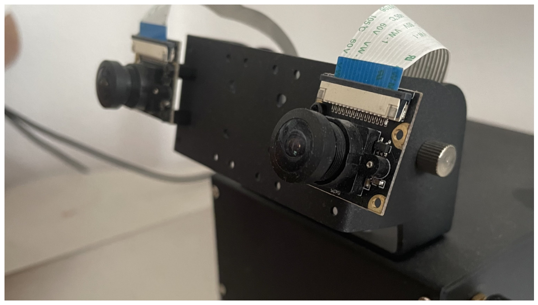


Fig. 2: Stereo Vision System implemented in this work

Fig. 3 represents the schematic for a simple Stereo Vision system. The goal is to find the z coordinate of the real-world point P. The two cameras have a focal length f and are separated by a baseline b. U_L and U_R represent the horizontal projection of P in the left and right image frames, respectively.

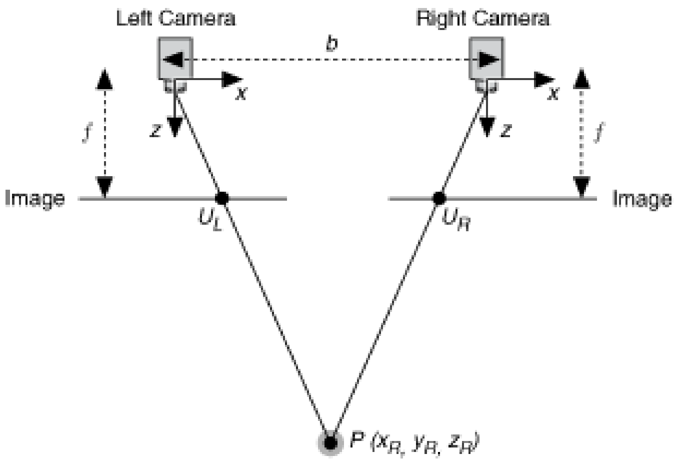


Fig. 3: Stereo Vision System Schematic [Retrieved from National Instruments]

Based on triangulation, there's a simple relationship between object P's depth and the disparity between U_L and U_R [12]. Using Eq. 1, it was possible to obtain the depth z for each keypoint detected by MediaPipe.

$$z = f\frac{b}{(U_L - U_R)} \tag{1}$$

B. Pose Detection and Angles Estimation

The task of pose estimation was carried out with the use of the MediaPipe Pose framework. MediaPipe Pose allows to perform Pose Estimation in real time by detecting and tracking body landmarks from images and video sources [13]. It provides the vertical and horizontal location of 33 body keypoints, i.e., x and y coordinates. MediaPipe can also estimate and provide the z coordinate for each keypoint, however, it was found that, for Yoga poses in particular, MediaPipe was not able to accurately estimate keypoints' depth. Therefore, to accurately perform this task, the Stereo Vision System was employed.

The combination of MediaPipe and the Stereo Vision System allowed for accurate estimation of x, y, and z coordinates for several keypoints of the body. With these coordinates, it was possible to calculate several joint angles using the vector dot product (Eq. 2), which allows to obtain the angle between two n-dimensional vectors.

$$cos\theta = \frac{u \cdot v}{||u|| \cdot ||v||} \tag{2}$$

Table I lists the angles that were defined to be calculated in this work while Fig. 4 illustrates these angles.

	Joint		Joint
1	Left Shoulder	7	Left Knee
2	Right Shoulder	8	Right Knee
3	Left Elbow	9	Left Shoulder 2
4	Right Elbow	10	Right Shoulder 2
5	Left Hip	11	Left Hip 2
6	Right Hip	12	Right Hip 2

TABLE I: List of defined angles

Note that, in Fig. 4, only half of the angles from Table I are shown. For instance, only the left shoulder (1) is drawn, while the right shoulder (2) is not. Similarly, only the right knee (8) is drawn, while the left knee (7) is not.

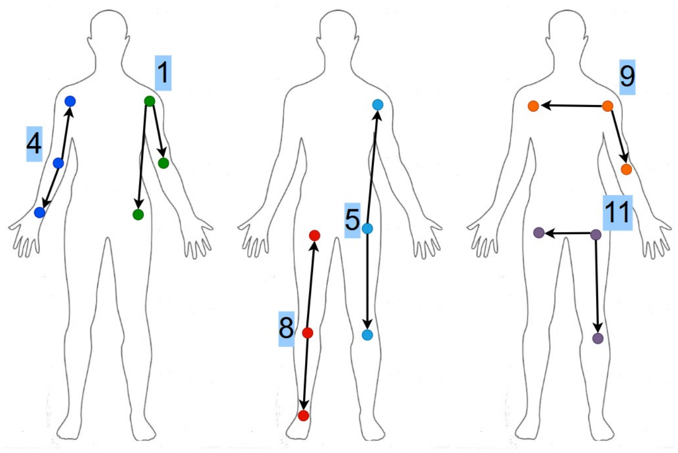


Fig. 4: Body angles

C. Validation of angles estimation with a Kinect

In order to validate the angle estimation accuracy of the Stereo Vision System, a Kinect V1 device was enabled. The justification of using the Kinect device was that it consists of a patented product and, at some point, was considered state of the art in terms of Pose Estimation. The Kinect device was programmed to also estimate the 12 previously described joint angles. It is important to mention that the Kinect device and Stereo Vision System operated simultaneously but independently while calculating joint angles during testing with volunteers.

D. Point-scoring system for pose evaluation

In order to evaluate pose accuracy, a point-scoring system was proposed in which the user is awarded points as they are performing a pose. The amount of points awarded depends on the magnitude of the percentage error between a particular joint angle of the user and its corresponding reference angle. In order to determine the reference angles for each pose, several Yoga images were downloaded from an open access dataset available online. Using MediaPipe, angle estimation was applied to all images in order to obtain average angle values for each pose, which served as the reference angles the user is compared to.

Eq. 3, which represents the percentage error, was used to compare the joint angles of the user with their corresponding reference angles. v_E represents the user's estimated angle while v_R stands for the reference angle. The magnitude of this error was used to determine the amount of points the user is awarded. This comparison is performed each time a pair of image frames are captured and processed using the Stereo Vision System.

$$\%Error = \frac{v_E - v_R}{v_R} \cdot 100\% \tag{3}$$

Fig. 5 shows an example in which the measured angle (θ) subjected to evaluation represents the elbow. If the error is relatively small, the user is awarded the highest amount of points.

As the error increases, the amount of points decreases. This process is performed for all 12 angles previously described.

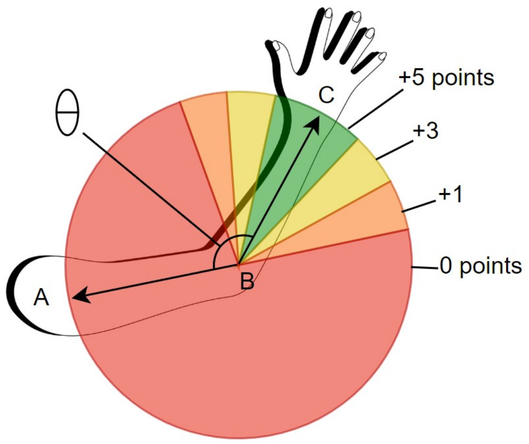


Fig. 5: Representation of the point-scoring system

Additionally, a color scale was defined in which colored circles are drawn on top of a stick figure that mimics the pose of the user. The goal of the color scale is to visually indicate the user which joints are correctly oriented in terms of the pose they are performing, and which are not. The color of a particular circle also depends on the magnitude of the percentage error, previously described, and follows the procedure of Fig. 5. The stick figure is drawn using the x and y keypoints coordinates provided by MediaPipe and is displayed next to the live video feed, on the Graphical Interface (GUI). An example is shown in Fig. 6.

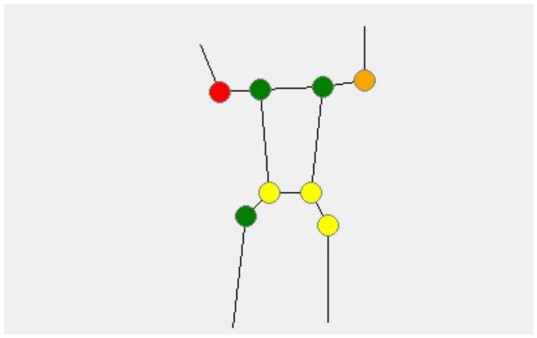


Fig. 6: Example of the stick figure and color scale system

Once all angles have been evaluated for a particular pair of frames, a global score is calculated by dividing the amount of points the user was awarded by the maximum amount of points they could have obtained. Since this procedure is constantly being performed, the global score is constantly being calculated (once for every pair of frames that is processed). A moving-average-like overall score is obtained by calculating the average between all global scores obtained for each pair of frames. The overall score value is also displayed on the Graphical Interface in order for the user to see it.

Fig. 7 represents the flowchart of the complete system. In order to achieve a higher video feed FPS count, two threads were defined to be executed in parallel. The first thread was used to display the live video feed from one of the cameras of the Stereo Vision System, while the second thread periodically retrieves a pair of frames from the first thread to run pose evaluation. Once the Yoga exercise is completed, both threads are terminated.

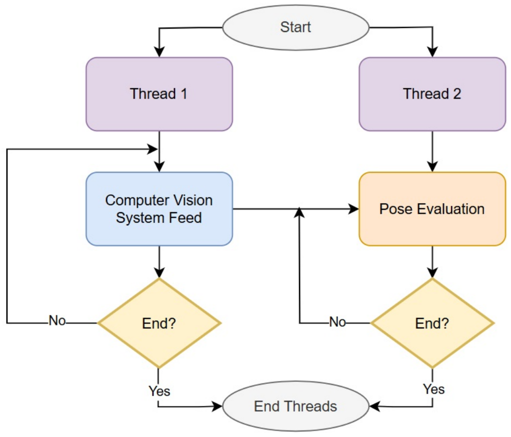


Fig. 7: Flowchart of the complete methodology process

IV. Results

The results of this work focus on corroborating that the Stereo Vision System is capable of accurately calculating joints angles. As it was described above, this was done with aid of the Kinect device. The first part of this section focuses on the obtained angles with both the Stereo Vision as well as Kinect systems, for each pose applied to this work. Additionally, an explanation regarding the designed Graphical User Interface is given, as well as the full hardware implementation for this work.

A set of tests was carried out with four volunteers where each of them was asked to perform each pose (i.e. Goddess, Warrior II, Tree) once for a few seconds while both the Stereo Vision System and Kinect were recording data.

As it was mentioned, the set of angles obtained by the Kinect device were taken as reference. It was observed that the Stereo Vision System was able to yield results with an error of 7% with respect to the Kinect.

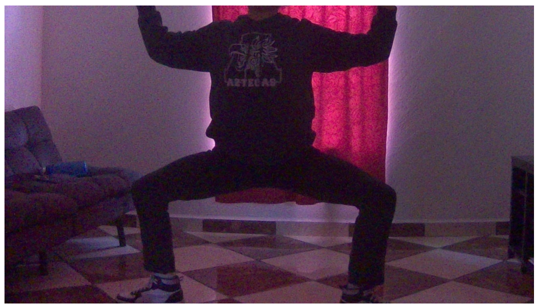


Fig. 8: First volunteer performing Goddess Pose

A. *Goddess Pose*

The first section of the testing stage consisted in the evaluation of the Goddess pose. An example is shown in Fig. 8 in which a volunteer is performing said pose.

The results for the obtained angles, using both the Stereo Vision and Kinect systems, are shown in Fig. 9 in the form of a radar chart. The chart's axis represents the value of the angle in degrees while each variable represents a particular joint.

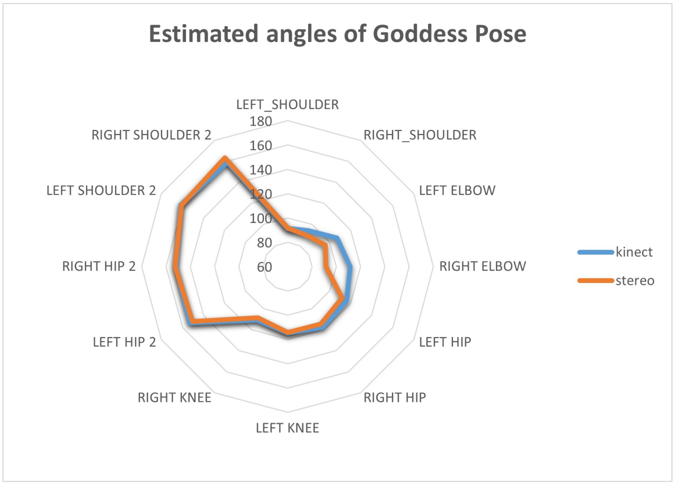


Fig. 9: Estimated angles for the Goddess Pose

B. *Warrior II Pose*

Subsequently, volunteers were asked to perform the Warrior II pose, as it can be seen in Fig. 10.

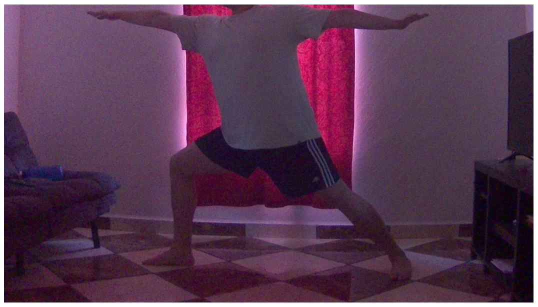


Fig. 10: Second volunteer performing Warrior II Pose

Similarly, a radar chart, shown in Fig. 11, was generated with recorded data from the Stereo Vision System and Kinect.

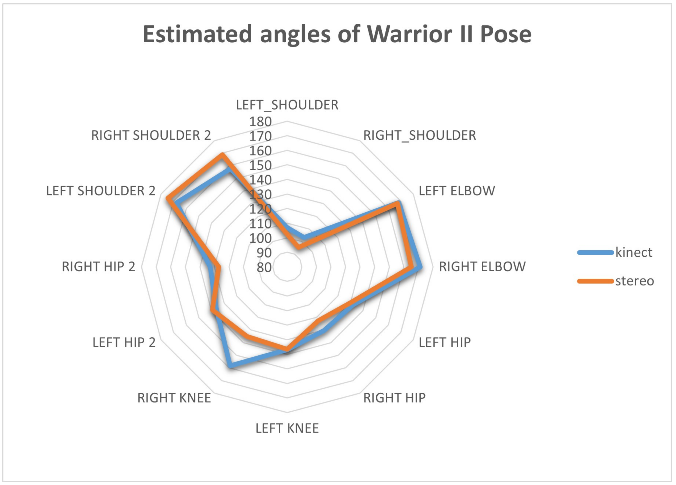


Fig. 11: Estimated angles for the Warrior II Pose

C. *Tree Pose*

Lastly, volunteers performed the Tree pose. Another volunteer can be seen in Fig. 12.

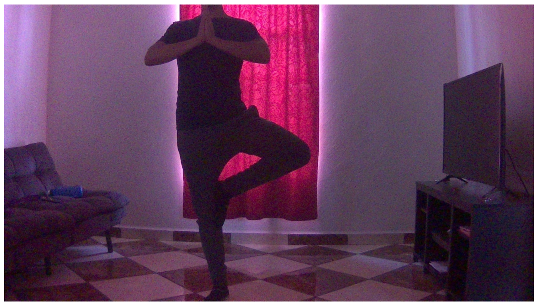


Fig. 12: Third volunteer performing Tree Pose

The results for the obtained estimated angles can be observed in Fig. 13.

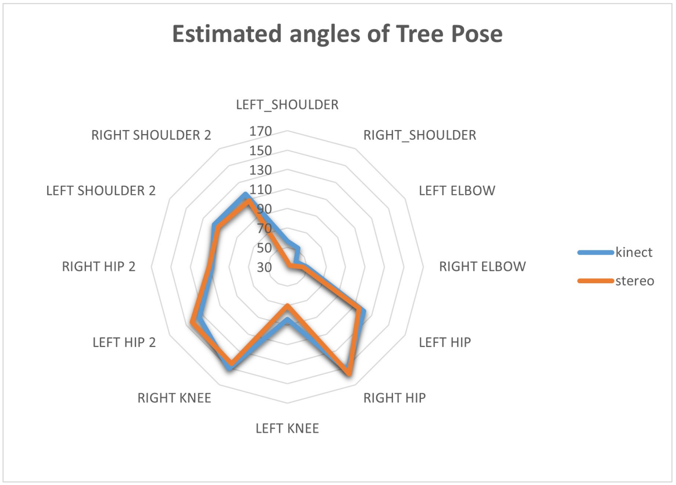


Fig. 13: Estimated angles for the Tree Pose

D. Graphical User Interface

As it was mentioned earlier, a Graphical User Interface was designed using the Tkinter toolkit for Python. As it can be observed in Fig. 14, the GUI allows the user to select the desired exercise, as well as the amount of time they'd like to perform it. The GUI also allows to select a difficulty level, ”Easy”, ”Medium” and ”Hard”. Depending on the selected difficulty level, the angle error tolerances, such as those shown in Fig. 5 are increased or decreased. Additionally, the GUI integrates a ”Start” button, as well as an ”End” button, which can be triggered before the running time ends.

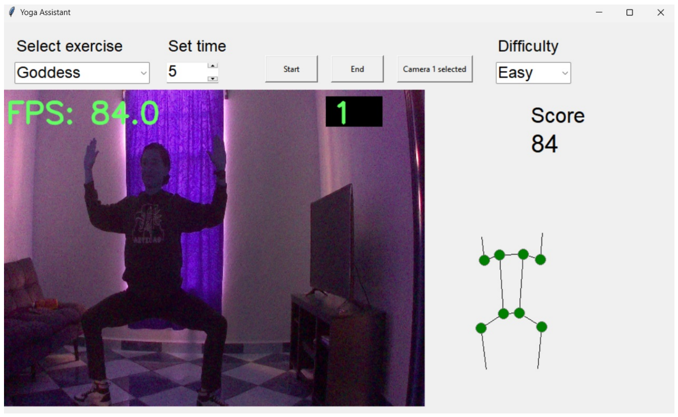


Fig. 14: Graphical User Interface

At the right of the GUI, the user is able to visualize their assigned score, which was previously described. Furthermore, they are able to see the stick figure and color scale system providing real-time feedback for the pose they are performing, such as the example shown in Fig. 6. Lastly, prior to beginning an exercise, the GUI displays an image of the selected exercise such that the user is given a reference of the correct way to perform said pose. Once the exercise begins, the live video feed from either of the IMX219 cameras is displayed in real time, instead of the reference image.

E. Full System Hosted on the Jetson Nano

The Jetson Nano Developer Kit is a compact computer manufactured by NVIDIA. It incorporates an NVIDIA GPU, which makes it the ideal platform to enable Artificial Intelligence, Robotics and Internet of Things applications [14].

The complete system was assembled as it is shown in Fig. 15. The two IMX219 cameras are connected to the Jetson Nano. The Jetson Nano lays inside the black case, which was acquired from Waveshare. For ease of use, a 7-inch touch screen was incorporated into the system. The touch screen allows users to interact with the system without the need for a mouse and keyboard.

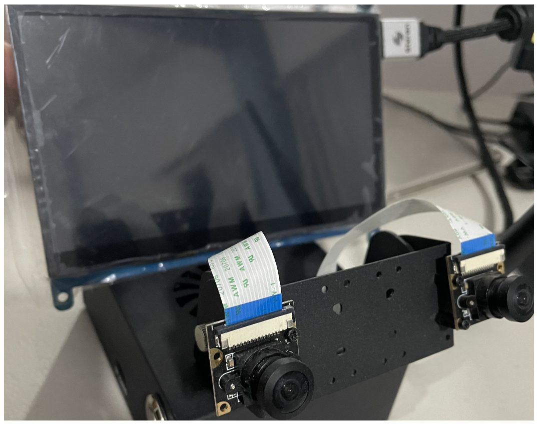


Fig. 15: Complete Stereo Vision System

V. Discussion and Future Work

A Stereo Vision System was implemented for the detection, evaluation and feedback of Yoga poses in real time. The system allows for interaction via a GUI in order to select the desired exercise, as well as additional parameters such as execution time and difficulty level. The GUI provides real-time feedback in the form of a 0-100 score, as well as a color-scale system to visually indicate how to improve.
With the aid of the Kinect device, it was possible to verify that the Stereo Vision System is capable of accurate estimation of several 3D joint angles of the human body, regardless of the chosen pose, as it can be seen in Figs. 9, 11 and 13. This demonstrates that the system is capable of accurately capturing the shape of the human body in terms of the joints that comprise it. The system is, therefore, capable of providing correct feedback as it is correctly detecting and estimating the human body.
This work also represents an improvement with respect to reviewed literature. While several works for the evaluation of Yoga poses have been developed, many of them only take into consideration x and y keypoints coordinates [6], [7], [9]. The developed Stereo Vision System allows for more accurate pose estimation and evaluation as it takes into account the depth coordinate for each keypoint.
The developed system will be of aid to users who perform Yoga, as it will provide accurate and correct feedback without the need of a certified instructor. This will reduce injury risk and instead help improve mental and physical health of individuals. Future work should focus on widening the available exercises to choose from. Additionally, the methodology followed in this work could be applied to further disciplines such as physiotherapy and weight training. Moreover, while this work focused on static exercises, in which users are only required to hold a pose for a certain amount of time, it can be extended to dynamic exercises.

Acknowledgments

The first author thanks the ”Programa de Maestría en Ciencias en la Especialidad de Electrónica” provided by INAOE, as well as the ”Programa de Becas para Estudios de Posgrado” provided by CONAHCYT.

References

[1] Luxmi Sharma. Benefits of yoga in sports –a study. *International Journal of Physical Education, Sports and Health IJPESH*, 1, 2015.

[2] Gyanesh Kumar Tiwari. Yoga and mental health: An underexplored relationship. volume 4, 2016.

[3] Christine Wiese, David Keil, Anne S. Rasmussen, and Rikke Olesen. Injury in yoga asana practice: Assessment of the risks. *Journal of Bodywork and Movement Therapies*, 23, 2019.

[4] Faisal Bin Ashraf, Muhammad Usama Islam, Md Rayhan Kabir, and Jasim Uddin. Yonet: A neural network for yoga pose classification. *SN Computer Science*, 4, 2023.

[5] Laia Turmo Vidal, Elena Márquez Segura, Luis Parrilla Bel, and Annika Waern. Training technology probes across fitness practices: Yoga, circus and weightlifting. Association for Computing Machinery, 4 2020.

[6] Maybel Chan Thar, Khine Zar Ne Winn, and Nobuo Funabiki. A proposal of yoga pose assessment method using pose detection for self-learning. 2019.

[7] Ayush Gupta and Ashok Jangid. Yoga pose detection and validation. 2021.

[8] Chhaihuoy Long, Eunhye Jo, and Yunyoung Nam. Development of a yoga posture coaching system using an interactive display based on transfer learning. *Journal of Supercomputing*, 78, 2022.

[9] Fazil Rishan, Binali De Silva, Sasmini Alawathugoda, Shakeel Nijabdeen, Lakmal Rupasinghe, and Chethana Liyanapathirana. Infinity yoga tutor: Yoga posture detection and correction system. 2020.

[10] Edwin W. Trejo and Peijiang Yuan. Recognition of yoga poses through an interactive system with kinect device. 2018.

[11] Rahul Kala. *Basics of Autonomous Vehicles*. 2016.

[12] Cha Zhang and Zhengyou Zhang. *Calibration between depth and color sensors for commodity depth cameras*, volume 67. 2014.

[13] Valentin Bazarevsky and Ivan Grishchenko. Google ai blog: On-device, real-time body pose tracking with mediapipe blazepose, 2020.

[14] NVIDIA Developer. Get started with jetson nano developer kit. https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit. Accessed: 2024-03-02.