Marker-less Augmented Reality Framework for Smart Teaching

Thesis submitted
in partial fulfillment of the requirements for the award of the

Master of Technology

in

Computer Science and Engineering

(under the Dual-Degree Programme)

by

Koniki Venkata Balaji
17CS02001

Under the supervision of
Dr. Debi Prosad Dogra

SCHOOL OF ELECTRICAL SCIENCES
INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

May 2022

APPROVAL OF THE VIVA-VOCE BOARD

May 10, 2022

Certified that the report entitled **"Marker-less Augmented Reality Framework for Smart Teaching"** submitted by **Koniki Venkata Balaji** (17CS02001) to the Indian Institute of Technology Bhubaneswar in partial fulfillment of the requirements for the award of the Master of Technology in Computer Science and Engineering under the Dual-Degree Programme has been accepted by the examiners during the viva-voce examination held today.

(Supervisor) **(External Examiner)**

(Internal Examiner 1) **(Internal Examiner 2)**

CERTIFICATE

This is to certify that the report entitled **"Marker-less Augmented Reality Framework for Smart Teaching"** submitted by **Koniki Venkata Balaji** (17CS02001) to the Indian Institute of Technology Bhubaneswar is a record of bonafide research work under my supervision and the report is submitted for end-semester evaluation of the Dual Degree thesis work.

(Supervisor)

DECLARATION

I certify that

1. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have followed the guidelines provided by the Institute in writing the thesis.
4. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
6. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Koniki Venkata Balaji
17CS02001

ACKNOWLEDGMENTS

I would like to express my profound gratitude to all the people who encouraged, motivated and helped me during my thesis work. First and foremost, I intend to thank my thesis supervisor, Dr Debi Prosad Dogra for his unparalleled support and motivation throughout the thesis work. Despite having a busy schedule, he constantly guided me with weekly review meetings and assessments. He made me feel comfortable with the challenges I faced during the course of research by proposing different probable solutions and practical approaches. The suggestions and insights he gave during our interactions were vital for bringing the best out of this work.

I would like to convey my regards to my research guide, Dr Ajaya Kumar Dash for his precious time spent making me familiar with AR concepts such as camera pose estimation, virtual object augmentation, etc. He was always available and willing to help me out. I greatly admire his commitment to his work in face of difficult times. I would also like to thank another research guide, Dr Shreetam Behera for the generous help that he offered with utmost passion. He was the person I consulted for any help in Machine Learning and Computer Vision fields. Without his passionate involvement and guidance, this work would not have been successful. I would also like to acknowledge the genuine efforts of a PhD Scholar, Mr Kamalakar Thakare in completing my thesis. He gave valuable tips for writing the thesis and also helped me in getting access to GPU resources for ML model training.

Finally, I like to express my love and gratitude to my parents and family members, who unfailingly supported me morally and emotionally. Without their care and backing, I would not have come this far. I would also thank all of my friends and batchmates at the IIT Bhubaneswar for making my five years of journey at the institute very memorable. I would cherish every moment spent with them that enhanced my academic knowledge and enriched my experience of life as a whole.

Koniki Venkata Balaji

ABSTRACT

In today's modern world, physical classrooms are getting transformed into smart classes at a rapid pace. In such a scenario, it is vital for both instructors and students to use the latest hand tracking and Human-Computer Interaction(HCI) technologies to explain and understand concepts more effectively. As there is a massive focus on developing the education sector, the use of modern technologies in a classroom setting is being given a lot of thought. Emerging technologies such as Augmented Reality(AR) and Virtual Reality(VR) have a great potential to enhance the teaching and learning processes.

Augmented reality is a technology that introduces artificial objects into the real-world scene and gives an interactive experience. Education can be made more interactive by using AR technology as it replaces memory-based rote learning with immersive experiences. This approach improves students' understanding of concepts and enhances their interest in learning. However, there is no real-time framework that augments a virtual object in the physical environment and facilitates bare human hand interaction with 3D objects.

This thesis proposes a framework that supports augmented reality with markerless human hand interaction features. It enables us to use specific hand gestures to perform operations such as zoom-in, zoom-out, rotation, etc., on the virtual objects. The framework makes use of Google MediaPipe Hands keypoint detection model for hand gesture recognition and OpenGL API for rendering 3D objects in the real-world scene. We have observed better performance of the MediaPipe Hands (mpHands) model in comparison with state-of-the-art object detection systems such as SSD-mobilenetV2, faster-Rcnn-inceptionV2 and r-FCN-resnet101. We have adapted the pre-trained object detection models (on MS-COCO 2018 dataset) to our problem by training them on a custom dataset. It has been observed that the mpHands model performed significantly better in terms of processing FPS.

Keywords : Hand Tracking, Augmented Reality, Human-Computer Interaction, Hand Gesture, Hand Keypoint, Object Detection

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The two technologies Augmented reality (AR) and virtual reality (VR) have the term "reality" in their names, although they vary in terms of whether or not they utilize the real - world environment. AR keeps the real physical world intact and adds digital content to the user's view in real-time [1], whereas VR indulges a user within a completely artificial world. In VR environment, the user's senses are aroused to give the feeling of immersion in a reality that is created virtually [2]. These technologies are changing the face of educational sector, by creating a number of practical applications that enhance teaching and learning processes. However, most of the use cases developed till now are dependent on specially designed hardware equipment which makes it challenging to deploy them in classrooms at a large scale. In this Thesis Work, we try to propose a classroom-learning framework based on AR technology that works on commodity hardware such as PC, laptop, etc. The chapter is structured as follows: The first section lists the overview of the area. While the second section talks about the motivations and objectives of the study, then the third section provides the main problem statement being solved in this work. Finally, the fourth section summarizes the proposed contributions.

1.1 Overview of the Area

Augmented reality (AR) is a technology that enables us to insert virtual objects in a real-world video feed effortlessly. A widely popular definition of augmented reality, *An AR system should combine real and virtual objects, be interactive in real-time, register real and virtual objects,* given by Azuma in the first ever survey on the subject [3]. AR and VR differ massively in terms of their features and end-goals. The AR technology enhances the real world by introducing an extra layer of virtual data, whereas VR works by taking away an user from the real-world through seamless immersion in an artificial 3D world. They also have a huge contrast in the way they offer the user experience.

AR can be used anywhere through mobile devices like smartphones, tablets, laptops, etc., or specially designed devices such as smart glasses. Users can continue daily activities while using AR glasses (interacting with the environment). AR is inherently connected to the surroundings, and its components, i.e, user and real-world agents. The technology depends on its real components to furnish extra details that enhance the user experience[4]. Here, users are able to see the environment around them, including the physical object present in surroundings and their own body [5]. Users do not get to experience any kind of strange isolation from the actual reality while using AR in contrast to VR. By using AR, users can render virtual entities directly into the real-world scene without any intermediary between them. Moreover, students can make use of AR technology in science tours, and in educational settings like laboratories and libraries for a more effective grasping of knowledge.

Both the technologies differ in many aspects, but they do share some properties

too. Both the technologies aid users to experience rich feelings that stimulate their senses, especially vision[6]. They expect engagement of users with both sensually and mentally. They enable an interesting possibility of interacting with digital content, motivating the users even more to be engaged with these technologies. None of them can work in place of the other. Each one has its role and fulfils specific needs. On one end, we have AR that augments the physical world with digital content. It needs a real-world scene as a continuous input feed for augmentation. Thus, AR technology becomes significant in the case of activities that focus on interpreting properties, features, or details of objects existing in the physical world. For example, the human body is a subject of study for anatomy students, artefacts arouse interest in archaeologists, and the virtual globe appeals better to the school students. On the other end, VR technology synthesizes a brand new artificial reality. Therefore, it is the perfect technology to experience events that are difficult or impossible to participate in current reality. Automobile engineering students can design vehicles and do simulations using VR. VR has a huge potential in the sports industry, where it can help players to train better by visualizing game scenarios virtually. It can also help viewers to experience a real stadium match from the comfort of their homes. Students can be taken on virtual field trips to the solar system, and different events of significance back in time.

1.2 Motivation and Objectives of the Study

AR based teaching in the education sector has evolved rapidly over the recent year. It has been shown that AR has the capacity to provide interesting learning experiences in classic classroom setting and also various knowledge spaces such as laboratories, museums, science centers, etc. It is said to have produced good results by enabling an

in-depth understanding of the concepts and better interaction between students and instructors. Therefore it is quite possible that the combination of interactive teaching framework and engaging digital content can help students remember what they have learned and develop skills by grasping information in no time. AR technology can be leveraged in most of the subjects taught to students of all age groups such as Biology, Geometry, Geology, etc. All the above said features and the potential of AR technology to transform the way of teaching are the motivations for this work.
The objectives of this work are:

1. To enhance classroom teaching methods with the usage of the latest technologies such as AR
2. To propose a methodology that makes learning more immersive and interesting
3. To develop a learning framework that is low-cost and uses fewer resources compared to commercial devices

1.3 Problem Statement

The problem that is being solved here is to design a novel framework in a classroom setup by making use of Augmented Reality and hand tracking technologies. The proposed framework should detect a bare human hand in the incoming video feed and augment 3D virtual objects on the hand after camera pose estimation in every frame. It should also support hand gesture recognition to perform basic interactions with the augmented objects such as zoom-in, zoom-out, and rotation. The problem also demands the solution be real-time on commodity devices for its wide range of implementation.

1.4 Proposed Contributions

In an attempt to solve the problem, we came up with a solution that uses the MediaPipe Hand Tracking Model, OpenCV, and OpenGL. The framework doesn't require any special hardware or extra hardware resources to function. It is designed to perform satisfactorily on a commodity PC consisting of a decent webcam. The contributions of the thesis are as follows:

1. A novel real-time working framework that augments 3D objects in input webcam feed and enables the user to interact with bare-hand gestures.

2. Comparison of mpHands model with different object detection models for hand gesture recognition.

3. Interaction with virtual objects such as zoom-in, zoom-out, and rotation around axes using simple human hand gestures without any markers.

Rest of the thesis is organized as follows. In Chapter 2, a brief survey of related work that has happened in recent years is provided. Next, the entire methodology of the proposed framework is presented in Chapter 3. The next chapter describes the experiments that were performed and tries to interpret the obtained results. Finally, concluding remarks and possible areas that we can work on in the future are presented in Chapter 5.

Chapter 2

Related work

2.1 AR Applications for Education

AR has a huge potential to enhance pedagogical practices due to its immersive experiences which make teaching and learning more interactive and interesting. As technology can impact the educational field at a massive scale, the application of AR technology in a classroom setting has been a greatly researched topic in the recent past since its emergence of AR technology. Augmented Reality has been defined broadly as two types: marker-based augmented reality and marker-less augmented reality. Marker-based AR works by augmenting objects on detection of an artefact like QR-code, symbolic photos, etc. The augmentation effect happens on scanning the marker present in the image frame. In the case of marker-less AR, the augmentation does not depend on any markers, instead, the system scans the real environment around the user and places the virtual objects in the scene.

Anatomy 4D is a marker-based AR application in which the marker image is the human body. The marker, once scanned, gets populated with digital information related to human anatomy. Similarly, star walk is a marker-less AR application in

which, just aiming the camera towards the sky results in a digital view of planetary objects within a certain range [7].

Similarly, work by Antonia et al. [8] analyzes the effectiveness of using AR contents in pre-school classes. They have shown the results were encouraging in the experimental group and AR technique is effective in learning modules for preschool students.

Maier et al [9] came up with the concept of "Augmented Chemical Reactions". Their work uses a checker physical cube as a marker on which virtual objects are augmented to perform a chemical reaction. A protein database was provided to choose the required molecules and this had a profound impact on the understanding of a subject.

Bower et al [10] review the use cases of AR in education and discuss the potential of technology in the teaching field. Quan Le et al. [11] proposed a learning framework to understand geometry concepts based on AR software and hand gesture recognition features.

Martin et al [12] proposed a learning system that focuses on objects in the field of industrial automation. The system is designed to use popular software tools on common mobile devices without the help of specialized equipment. They have used hybrid AR markers that carry details about the object of study and its orientation relative to the observer.

Barmaki et al [13] presented a learning framework using an education tool called REFLECT for premedical anatomy students. The system makes use of the AR magic mirror technique to augment digital visuals of human anatomy over the user's body in a large display present in front of the user. This creates an illusion of virtual anatomy inside the user's body. Their results show that the framework helped better understand the human musculoskeletal system.

2.2 Hand Pose Estimation

Hand pose estimation is a vital part of most of the applications where human-computer interaction is involved. This involves continuous hand-tracking in the incoming video feed from the camera and estimating the pose of the hand skeleton in terms of hand keypoints, finger joints, etc. Hand tracking is a significant aspect of a natural user interface and communication in products that make use of AR, and VR technologies and a significant amount of research related to hand tracking has been carried out in the industry. [14] [15].

Vision-based hand pose estimation is one of the most researched areas in the recent past. Most of the previous works require specialized hardware like depth sensors[16][17][18] which are not computationally compatible to perform in real-time on general inexpensive devices and therefore are confined to platforms that make use of heavy-duty processors.

Mueller et al [19] presented an approach for real-time hand pose estimation for first-person view RGB-D cameras in real environments with a lot of background distractions. Their approach uses two CNNs that localize and estimate the 3D joint locations of the hand. They have created a benchmark dataset *EgoDexter* and the approach has performed with low error consistently under difficult occlusions and cluttered background fluctuations.

A mobile application is presented by Gruosso et al [20] that does 2D hand tracking from the RGB camera of a smartphone. They have decreased the computational times for processing images on RegNet because of fewer resources on mobiles by tweaking a few components of the network. They achieved a response time of 0.05 seconds per frame, i.e, 18fps on a device with Qualcomm Snapdragon 855 SoC.

A two-stage convolutional neural network architecture was presented by wang et

al [21] for 2D hand pose estimation with 21 key points from in-the-wild RGB images. The pipeline has a mask prediction state and a pose estimation stage. They have contributed a new hand dataset OneHand10K, where each image is labelled with a segmented mask and key points.

Seo et al [22] proposed an AR system that supports one-handed augmentation and bare hand interaction on mobile devices. They devised a palm pose estimation method that makes use of four points estimated from the natural features of the hand. The framework also allows the user to interact with the object by detecting hand motion using the fingertip tracking method.

Instead of two-stage pipelines that localize the hand and detect the key points, Li et al [23] proposed an effective single network, an FCN trained end-to-end, working on a pose anchor network, an extension of RPN in Faster-RCNN. The pose anchor is produced by a K-means clustering technique on object keypoint similarity. With the usage of this pose anchor, the network can partially solve the occlusion challenge by processing past knowledge of hand structure.

In this report, we propose a teaching model based on a Real-time Hand Tracking solution designed by MediaPipe, Google [24]. We used the aforementioned model from MediaPipe to design a teaching framework that enables a user to augment virtual objects on hand and interact with them using hand gestures.

2.3 Object Detection Techniques

The task of object detection is carried out largely in two ways, traditional machine learning and deep learning methods. For tradition methods, feature engineering need to be done manually which is done majorly in three kinds. They are General Hough transform [25], Harris corner detector [26] and SIFT algorithm [27].

On the other hand, Deep learning methods don't require effective feature engineering and produce better results than traditional counterparts when trained on large datasets. Different object detection algorithms based on deep learning can be broadly grouped into three categories, SSD, Region and YOLO-based models.

Girshick et al [28] proposed a scalable object detection algorithm R-CNN that runs a CNN on top of 2000 region proposals generated by selective search. The outputs of CNN are fed into an SVM to classify the detected region and linear regression is done over the bounding box coordinates of the detected object.

As an improvement to the R-CNN model, the author came up with another model called Fast R-CNN [29]. The model improved over its predecessor by extracting CNN features before proposing regions and replacing SVM component with a softmax layer at the end.

A better model called Faster-RCNN [30] was proposed which replaced slow selective search algorithm with a fast neural network called Region proposal network(RPN). The region proposals from the RPN are fed into typical Fast R-CNN.

Dai et al [31] proposed R-FCN, region based full convolutional networks to improve the efficiency by avoiding per-region computations multiple times. It is fully convolutional with all computations shared across the image. It makes use of position-sensitive score maps that represent relative positions of each object class.

Liu et al [32] presented an algorithm SSD, Single-Shot Detector, to detect objects in images using a single deep neural network, i.e, simultaneously predicting the bounding box and class in a single shot. The model uses two techniques, non-maximum suppression and hard negative mining to balance the classes during training. This model enjoy enormous speed gains over region based models.

Chapter 3

Proposed Methodology

The framework design is explained in three sections. Section 3.1 describes how the MediaPipe solution extracts 21 hand key point coordinates in the bare human palm from an input video frame. Section 3.2 talks about the theory behind camera pose estimation and how it is done. Finally, Section 3.3 explains how virtual objects are registered in real-world scenes, and describes all the gestures that are used to interact with them. The overall framework is as shown below in Figure 3.1.

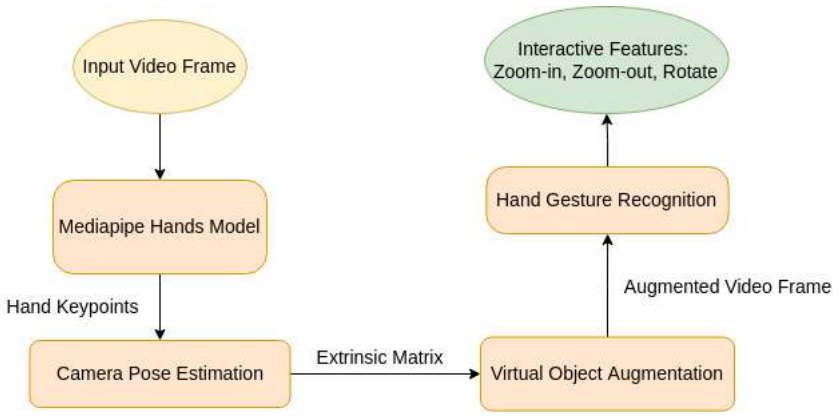


Figure 3.1: Framework Overview

3.1 Hand Landmark Extraction

The Hand Tracking pipeline presented in [24] consists of two models:

3.1.1 BlazePalm Detector

A palm detector was trained to estimate bounding boxes of palms and fists which is comparatively easier than detecting the entire hand in different orientations. Palms were represented using square bounding boxes. An encoder-decoder feature was used for a larger scene-context awareness.

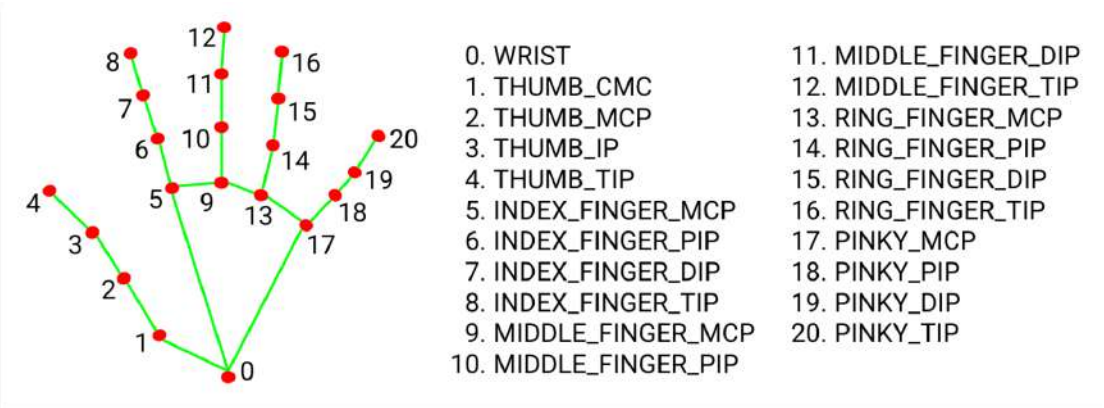


Figure 3.2: 21 Landmark Points detected by MediaPipe Hand Tracking Solution

3.1.2 Hand Landmark Model

The Hand Landmark model detects hand skeleton by localizing 21 2.5D coordinates in the detected bounding boxes of palms from the previous model using regression as shown in Figure 3.2. The model consistently learns the hand pose in subsequent frames and is also robust to challenges such as partial visibility and occlusions. The model has the following outputs:

- Set of 21 hand keypoints with x, y coordinates, and z relative depth

- A probability score for presence of hand in the input image
- Binary value indicating right or left-handedness for each detected hand

3.2 Camera Pose Estimation

This task is the most crucial one for the performance of the entire application. This subsection deals with the challenge of determining the location and orientation of the camera with respective to the object. We use the correspondence between 2D image coordinates and 3D object positions to compute the pose. 3D world coordinates are transformed into 2D image coordinates using the camera perspective transformation formula. A brief overview flowchart of pose estimation can be seen in Figure 3.3. This entire process of obtaining approximate values of different camera parameters is called camera calibration.

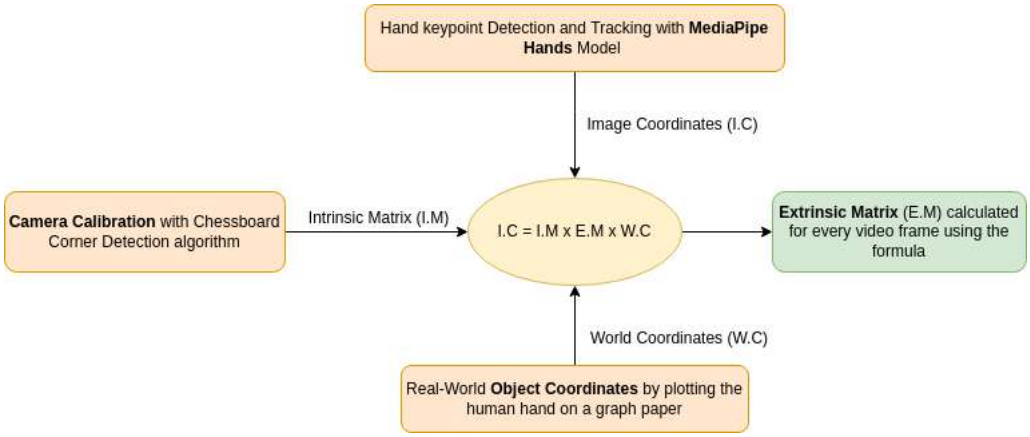


Figure 3.3: Camera Pose Estimation

Camera calibration involves estimating all the camera parameters. This means obtaining all the information about the camera required to successfully project a 3D object point in the real world into corresponding 2D image pixel in the image captured by that particular camera.

This constitutes obtaining two kinds of parameters:

1. **Intrinsic parameters:** These values represent the inherent features of the camera system that are constant by default for a camera piece.E.g. optical centre, focal length, etc.

2. **Extrinsic parameters:** The set of parameters that gives the orientation of the camera with respect to a world coordinate system represented in terms of rotation and translation vectors.

The equations that relate 3D object position vector (X_w, Y_w, Z_w) in world coordinate system to its projection (u, v) in the image coordinate system are shown in Equations 3.1, Where $\mathbf{P}$ is a 3x4 Projection matrix consisting of two components - intrinsic matrix($\mathbf{K}$) and extrinsic matrix($[\mathbf{R} \mid \mathbf{t}]$), an augmented matrix of 3x3 rotation matrix $\mathbf{R}$ and 3x1 translation $\mathbf{t}$ vector as given in Equation 3.2.

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.1}$$

$$u = \frac{u'}{w'} \qquad v = \frac{v'}{w'}$$

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}} \tag{3.2}$$

The intrinsic matrix $\mathbf{K}$ shown in Equation 3.3 is upper triangular consisting of x and y focal lengths (f_x, f_y), coordinates of optical center (c_x, c_y), and skew γ between the axis which is usually 0.

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The camera's extrinsic matrix in Equation 3.4 tells about the camera's position and orientation in the world coordinate system. It is composed of two components: a rotation matrix, **R**, and a translation vector **t**.

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right] \tag{3.4}$$

3.2.1 Intrinsic Matrix Calculation

The intrinsic matrix of the camera which represents its inherent features is approximated using the function *cv2.cameraCalibrate* available in the OpenCV library. Initially, nearly 10 checkerboard images are taken in different orientations and corners are detected using the *cv2.findChessboardCorners* available in OpenCV. The coordinates of corners are further refined with sub-pixel level precision by using a function *cv2.cornerSubPix*. Then, the calibrating function takes image and object points of the checkerboard corners as input and gives an output which includes intrinsic matrix and distortion coefficient matrix. These two matrices are used in subsequent sections.

3.2.2 Extrinsic Matrix Calculation

The main objective of this task is to calculate the extrinsic matrix of the camera for every video frame. It gives the estimated orientation of the camera w.r.t the virtual object. six specific hand key points among the detected are used as hand features to

estimate the camera pose. Their image coordinates are provided by hand detection model output and object coordinates are obtained by plotting the hand on a graph paper.

Intrinsic parameters are estimated as given in the previous subsection and extrinsic parameters are obtained by finding the unknown in the projection formula. The entire flow of camera pose estimation is described in the flowchart Figure 3.3. The estimation of the extrinsic matrix using all the known values is done with the help of the function *cv2.solvePnP*. It takes object and image coordinates, intrinsic matrix and distortion coefficient matrices as input, producing an output having rvec and tvec that constitute the extrinsic matrix.

3.3 Virtual Object Interaction

After obtaining all camera parameters, virtual objects are augmented on the required location using the same projection formula that was discussed earlier. As the camera transformation matrix projects the 3D world coordinates on to Image plane, the virtual object augments into the real world scene environment. Initially, to see the stability of augmentation, simple figures such as 3D axis and cube were registered using only the OpenCV functions. After getting stable outputs, we have implemented the augmentation in the OpenGL environment which enables us to visualize complex 3D objects. The **OpenGL API** is implemented using the **Glumpy** package which offers wrapper functions that use raw OpenGL under the hood. This section explains the process in two parts, firstly virtual object augmentation and then interaction with hand gestures.

3.3.1 Object Augmentation

Six hand keypoints having indices 0, 1, 5, 9, 13, and 17 according to the hand skeleton in Figure 3.2 are selected as image points to generate a coordinate plane on the palm. These points are preferred because they stay relatively static most of the times irrespective of hand and finger movements. In an OpenGL environment, object rendering is always relative to the camera. Therefore, virtual object vertices should be represented with the camera as a reference. Model, View and Projection are a chain of matrices that are used to translate a vertex represented in model space into image space before rendering. The three matrices perform transformations from one space to another as explained below.

1) Model Matrix:

A model matrix **M** is constitutes the object's translation, rotation and scale transforms **T**, **R**, and **S** respectively. Multiplying it with a vertex position vector v transforms the vector into world space representation as shown in Equation 3.5.

$$\begin{aligned} \mathbf{M} &= \mathbf{T} \cdot \mathbf{R} \cdot \mathbf{S} \\ v_{world} &= \mathbf{M} \cdot v_{model} \end{aligned} \tag{3.5}$$

2) View Matrix:

As all renderings are done from a camera's view, object vertices need to be represented taking the camera as reference. *Camera space* is a coordinate system, where the camera is situated at the origin position (0, 0, 0), directed along the negative Z axis. Similar to virtual objects, the camera as well possesses a model matrix that represents its location in the world space. The inverse of this model matrix is nothing but view matrix, and it converts vertices represented in *world space* to corresponding

vertices in *camera space* or otherwise called as *view space* as shown in Equation 3.6.

$$v_{camera} = \mathbf{V} \cdot \mathbf{M} \cdot v_{model} \tag{3.6}$$

3) Projection Matrix:

The projection matrix controls the portion of the scene to be captured while rendering by setting the extent of the camera's view. The two popular projections are *perspective* and *orthographic*. After multiplying with the projection matrix, the object's vertices are represented in *clip space* as given in Equation 3.7.

$$v_{clip} = \mathbf{P} \cdot \mathbf{V} \cdot \mathbf{M} \cdot v_{model} \tag{3.7}$$

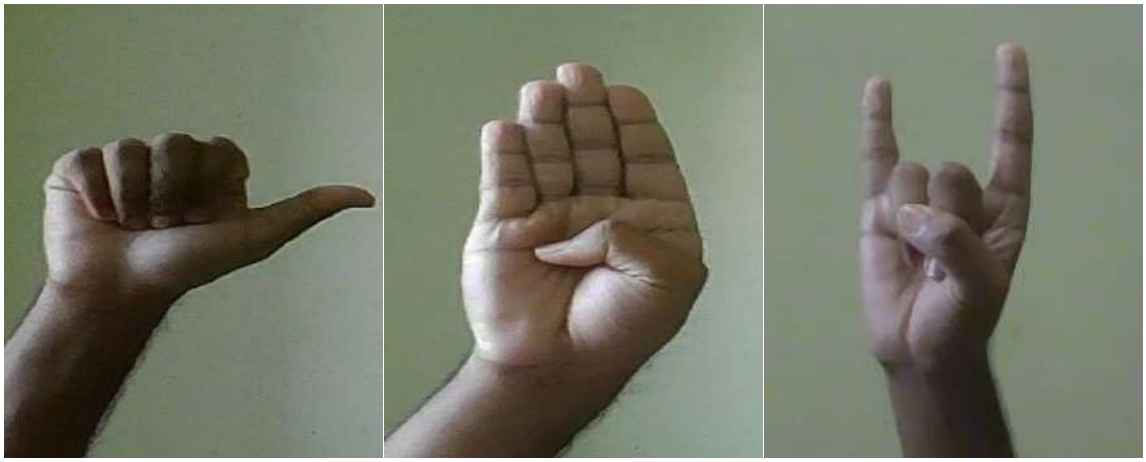
By multiplying a position vector by the chain of model, view, and projection matrices, vertices initially defined in *model space* are transformed into *clip space* via *world space* and *camera space*. The coordinates of vertices should be between -1.0 and 1.0 to be visible in the rendered window. Therefore, all the vertices are normalized to the -1.0 and 1.0 range by perspective division which takes into account the range of space we want to render. Finally, the normalized coordinates are processed by the rasterizer to produce image 2D coordinates or pixels on the screen.

3.3.2 Gesture Interaction

Once the object is rendered in the scene, hand gesture based interactions with the object is performed such as registering, zooming, rotating and deregistering. Six comfortable gestures are formulated for six different tasks which can be detected by comparing specific x, and y coordinates of hand keypoints detected by the Mediapipe model.

To make the zooming interaction more interactive, dynamic gestures are implemented. This is achieved by maintaining a queue of fixed size to store distances between the tips of the index and thumb fingers. Zoom-in and Zoom-out scaling are performed only when all the values in the queue are ascending and descending in order respectively. Dynamic gestures enable us to change the size of the objects by standard pinch-to-zoom actions we do on touch screens. The six gestures used are as shown in Figure 3.4.

(a) Register (b) Zoom-In (c) Zoom-Out



(d) Rotate-X (e) Rotate-Y (f) Deregister

Figure 3.4: Hand Gestures

The six gestures function as follows:

1. **Register:** After rendering the 3D object in the scene by augmenting over the left-hand palm, making the Register gesture fixes the object's position in the space. Once the object is registered, the left hand can be taken out of the image frame.

2. **Zoom-In:** The virtual object expands in size by moving the thumb and index finger away in the Zoom-In gesture, i.e, making a pinching out movement.

3. **Zoom-Out:** The object compresses into smaller size by bringing the thumb and the pair of the index, and middle fingers together in the Zoom-Out gesture, i.e, making pinching in movement.

4. **Rotate-X:** Rotation of the object around its local X-axis is obtained by folding all fingers except the thumb finger pointing outwards horizontally.

5. **Rotate-Y:** Object can be made to rotate around the local Y-axis by extending all the fingers except the thumb which is folded into the palm.

6. **Deregister:** By doing the Deregister gesture, we can completely remove the object from the space by not rendering it in the first place.

Chapter 4

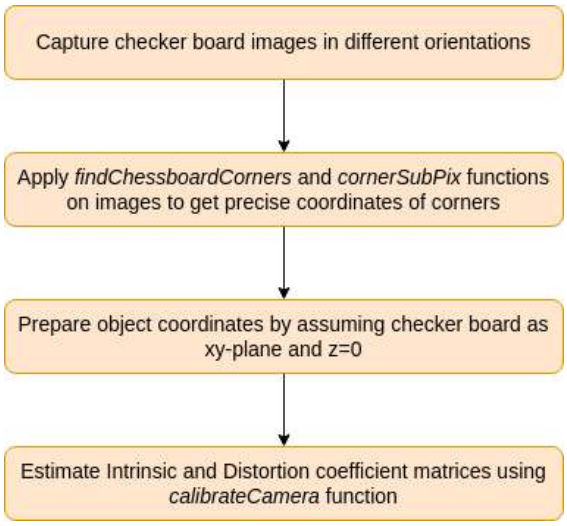
Experiments and Results

4.1 Camera Calibration

This experiment estimates the two components of the camera, i.e, Intrinsic and Extrinsic parameters as discussed in section 3.2. We have estimated the parameters of a laptop's built-in camera device (usb2.0 HD UVC webcam) for our project. We have shot 10 pics of a standard 7x9 checkerboard used for camera calibration. It has 20x20 mm squares printed on A4 paper. After doing multiple tasks as listed down in the flowchart 4.1a, we have obtained the intrinsic matrix 4.1b as a result.

After obtaining intrinsic parameters, object coordinates of our interest are defined by plotting the human palm on a graph paper 4.2b. The approximated real-world coordinates of hand features, images pixel coordinates and intrinsic parameters are the known values which are used to estimate the unknown, extrinsic matrix with the help of OpenCV functions as per the diagram 4.2a.

The camera pose estimation is validated by creating a simple 3D axis and cube over the checkerboard as shown in Figure 4.3. After getting an extrinsic matrix

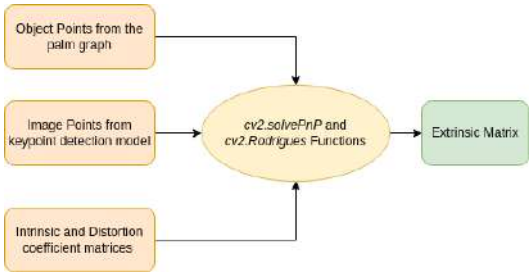


(a)

```
[[641.45028614   0.         310.9697875 ]
 [  0.         641.62230146 230.15924166]
 [  0.           0.           1.        ]]
```

(b)

Figure 4.1: Intrinsic Matrix calculation procedure and Result



(a)

(b)

Figure 4.2: Extrinsic Matrix calculation and Palm graph

for every frame, the camera transformation formula is used to display 3D figures at desired spots on the checkerboard which is the world coordinate system. x,y coordinates in the image are obtained from homogenous coordinates resulting after matrix multiplication of camera matrix and pose vector in the world space.

(a) 3D-axis (b) 3D-cube

Figure 4.3: Simple Augmentations over Checkerboard using OpenCV

4.2 Augmentation Outputs

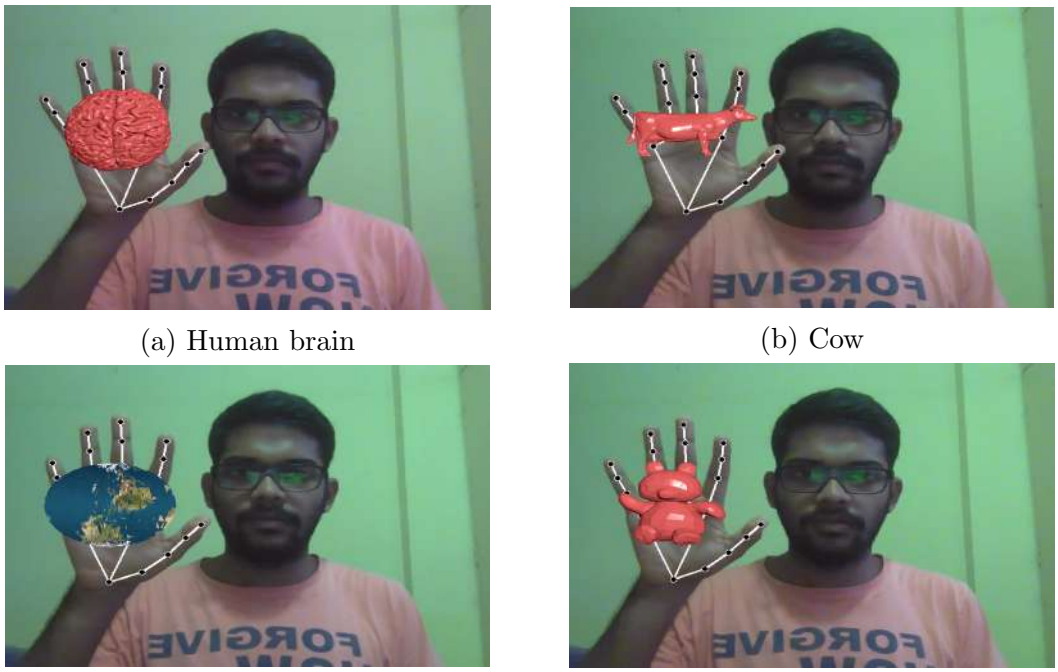
Initially, augmentation is carried out in an exclusively OpenCV environment, and the results are stable and satisfactory. The OpenCV environment makes use of projected 2D coordinates obtained from homogenous coordinates after camera transformation. These 2D coordinates are used to draw lines using OpenCV functions. 3D axis and cube are augmented on hand as shown in Figure 4.4.

(a) 3D Axis (b) 3D Cube

Figure 4.4: Augmentation using OpenCV

After getting decent results in the OpenCV environment, we started to implement the same using OpenGL API which can render more complex 3D virtual objects. As done before, camera parameters are calculated and then projection, and view matrices

are constructed from intrinsic and extrinsic matrices respectively. Initially, the model matrix is created in such a way that centre of the object is positioned on the palm. Few virtual objects such as the human brain, cow, teddy, and globe are rendered on hand as shown in Figure 4.5.

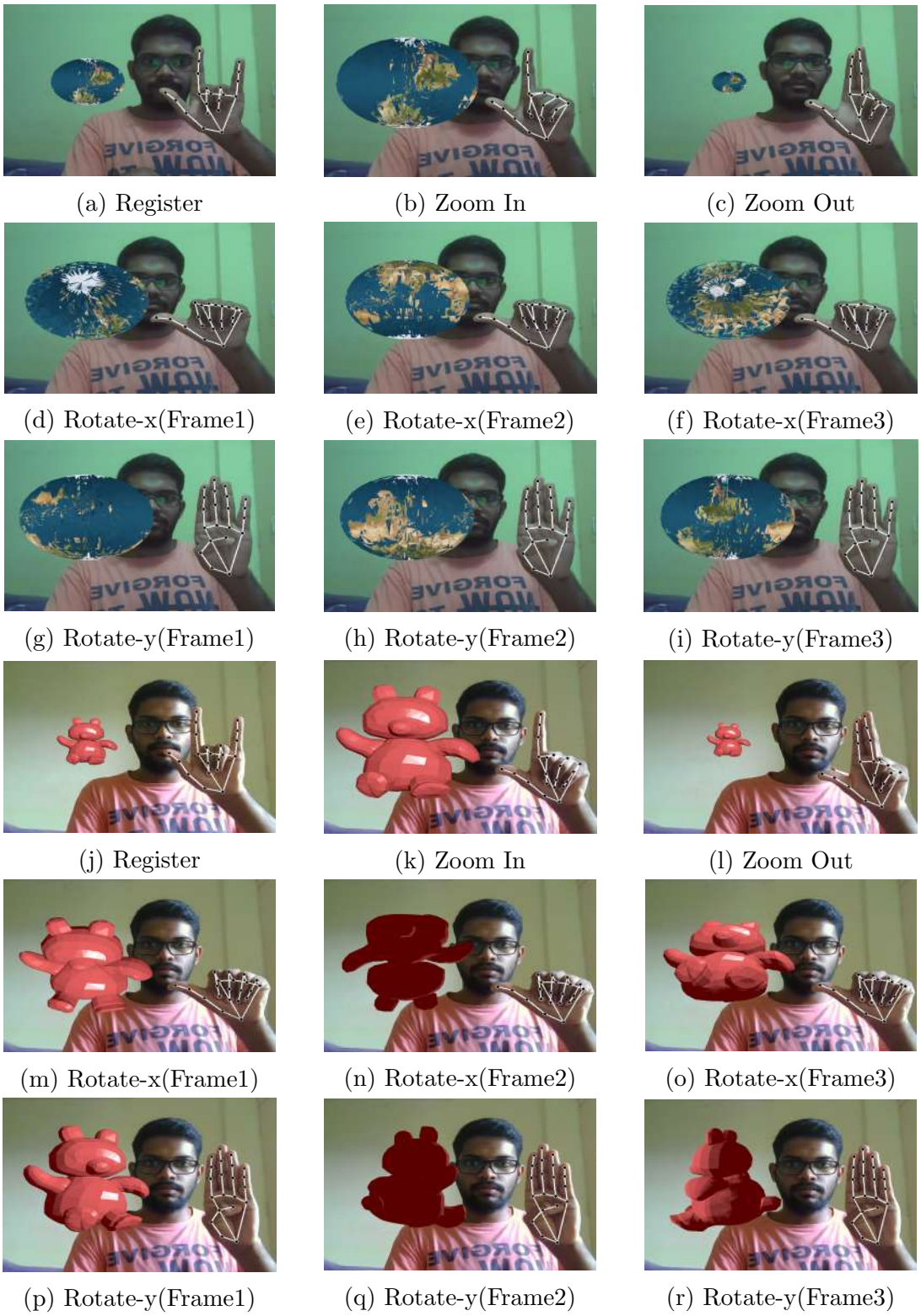


(a) Human brain (b) Cow

(c) Globe (d) Teddy

Figure 4.5: Augmentation of 3D Objects using OpenGL

Different hand gestures are used to interact with the objects in various ways such as fixing the object at a point in space, zooming in and out, rotating around local x, y-axes and removing the rendered object. Varying the size of the object is done by using of scale matrix which is a part of the model matrix of the object using *glm.scale* function. Rotating actions are carried out by using *glm.rotate* function. Interactions with the virtual objects are shown in Figures 4.6, 4.7. Each object has 3 rows of images, first-row consisting of register, zoom-in, and zoom-out outputs, and the next two rows consisting of outputs of rotations around the x, and y axes taken at three different times for each axis.



(a) Register (b) Zoom In (c) Zoom Out

(d) Rotate-x(Frame1) (e) Rotate-x(Frame2) (f) Rotate-x(Frame3)

(g) Rotate-y(Frame1) (h) Rotate-y(Frame2) (i) Rotate-y(Frame3)

(j) Register (k) Zoom In (l) Zoom Out

(m) Rotate-x(Frame1) (n) Rotate-x(Frame2) (o) Rotate-x(Frame3)

(p) Rotate-y(Frame1) (q) Rotate-y(Frame2) (r) Rotate-y(Frame3)

Figure 4.6: Augmentation of virtual objects Globe and Teddy

(a) Register (b) Zoom In (c) Zoom Out

(d) Rotate-x(Frame1) (e) Rotate-x(Frame2) (f) Rotate-x(Frame3)

(g) Rotate-y(Frame1) (h) Rotate-y(Frame2) (i) Rotate-y(Frame3)

(j) Register (k) Zoom In (l) Zoom Out

(m) Rotate-x(Frame1) (n) Rotate-x(Frame2) (o) Rotate-x(Frame3)

(p) Rotate-y(Frame1) (q) Rotate-y(Frame2) (r) Rotate-y(Frame3)

Figure 4.7: Augmentation of virtual objects Human brain and Cow

4.3 Object Detection Model Training

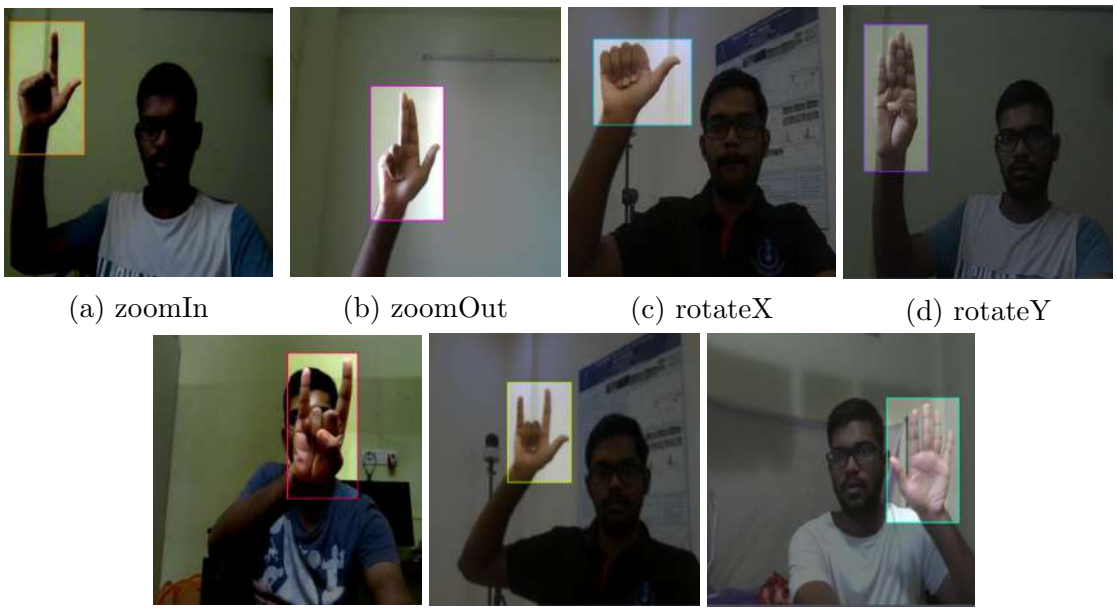
To use the best components in the framework, we have explored other alternatives for hand gesture recognition apart from the used keypoint detection model. We have used three state-of-the-art object detection models, SSD + Mobilenet V2 [32][33], Faster R-CNN + Inception V2 [30] [34] and R-FCN + Resnet 101 [31] [35], with pre-trained weights using Tensorflow object detection API. The three models are fine-tuned by training on a new dataset for our gesture recognition problem.

4.3.1 Dataset Preparation

The image dataset is created by capturing images of all hand gestures in different backgrounds and lighting conditions. Apart from the actual 6 gestures, an extra gesture named *leftPalm* is included. It is used to render the object initially before performing an interaction. The names of seven gestures and their interactions are given below:

1. zoomIn - Expands the object
2. zoomOut - Compresses the object
3. rotateX - Rotates the object around its local X-axis
4. rotateY - Rotates the object around its local Y-axis
5. fixPos - Registers the object in the world space
6. clear - Removes the rendering of the object
7. leftPalm - Renders the object on the palm of left hand

As a whole, 532 images are captured equally distributed per gesture. The captured images are labelled by using *labelImg* annotation tool. This set of images is further scaled by doing data augmentation tasks such as changing image brightness, contrast and sharpness. After data augmentation, a dataset consisting of 2660 images is divided into training, validation and test datasets as per a 70-20-10 split, i.e, 1862, 532 and 266 images. Few samples of the final dataset are displayed in the figure 4.8. The image dataset is converted into *TFRecord* format as per Tensorflow training standard with the help of Roboflow software.



(a) zoomIn (b) zoomOut (c) rotateX (d) rotateY

(e) clear (f) fixPos (g) leftPalm

Figure 4.8: Image Dataset Samples

4.3.2 Model Training Results

By using the Transfer learning technique, we have trained the three models as per details provided in the table 4.1. For training, we have utilized the tutorial notebooks provided by Roboflow for training object detection models on custom datasets. The

loss curves of the models over the course of training are accessed through the tensorboard package and displayed in figure 4.9. Each row has graphs of validation loss, training loss and mAP@0.75IoU. Table rows from top to bottom represent graphs for SSD, Faster R-CNN and R-FCN respectively. It can be observed from the curves that a major portion of loss decrement has happened within the first 2000 steps for the R-FCN model.

Model	**Input size**	**Optimizer**	**Steps**	**Batch-size**	**Learning-Rate**
SSD MobilenetV2	300x300	RMSProp	$20,000$	12	0.004
Faster R-CNN InceptionV2	600x1024	Momentum	$20,000$	12	0.0002
R-FCN Resnet101	600x1024	Momentum	$20,000$	8	0.0003

Table 4.1: Model Training Details

After training the models, an inference test has been performed on the test dataset which resulted in accuracy metrics as shown in the table 4.2. As per the obtained results, it is observed that all the models are performing significantly good at IoU scores of 0.50 and 0.75. When we look at the prime metric, i.e, AP @ IoU = 0.50:0.05:0.95, descending order of accuracy is SSD, Faster-RCNN and R-FCN. The metrics used to measure accuracy are described below:

1. AP - Average Precision (Average value of area under Precision-Recall curves)
2. AR - Average Recall (Average value of twice the area under Recall-IoU curves)
3. IoU - Intersection over Union (Area of overlap / Area of Union)
4. area - Area of detection box in square pixels
5. small - $area < 32^2$
6. medium - $32^2 < area < 96^2$

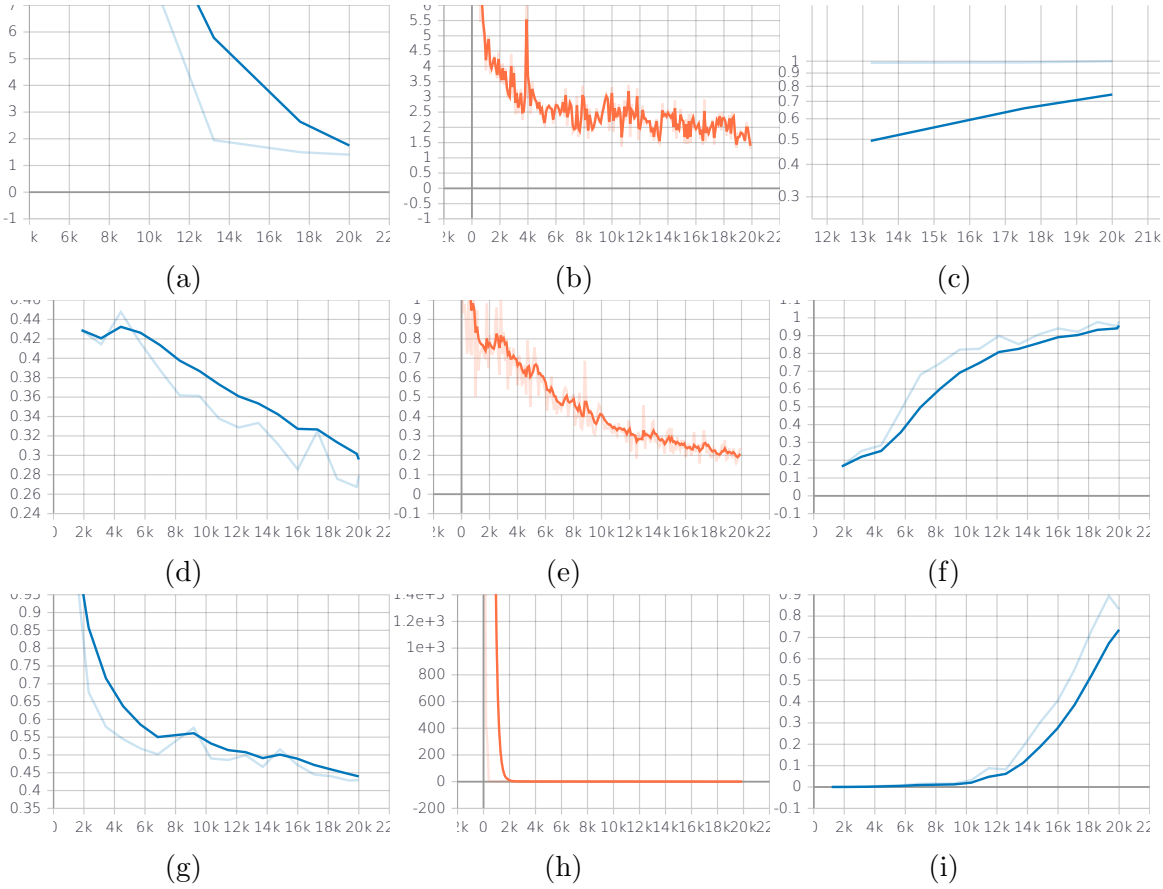


Figure 4.9: Object Detection Models Loss and Precision Curves

7. large - $area > 96^2$

8. maxDets - Maximum detections per image

4.4 Comparison Results

we have compared the inference results of object detection models with the keypoint detection model by calculating its accuracy metrics over the same test dataset used before. Each image in the dataset is given as input to the mpHands model and hand gesture is predicted by comparing the coordinate values of different key points of the detected hand skeleton. As a result, we have two vectors of truth values and

Accuracy Metric	**SSD**	**Faster-RCNN**	**R-FCN**
AP @[IoU=0.50:0.95 | area=all | maxDets=100]	0.829	0.755	0.666
AP @[IoU=0.50 | area=all | maxDets=100]	0.999	0.997	0.966
AP @[IoU=0.75 | area=all | maxDets=100]	0.999	0.971	0.860
AP @[IoU=0.50:0.95 | area=small | maxDets=100]	−1.000	−1.000	−1.000
AP @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.800	0.700	0.755
AP @[IoU=0.50:0.95 | area=large | maxDets=100]	0.830	0.755	0.667
AR @[IoU=0.50:0.95 | area=all | maxDets=1]	0.853	0.799	0.729
AR @[IoU=0.50:0.95 | area=all | maxDets=10]	0.853	0.800	0.740
AR @[IoU=0.50:0.95 | area=all | maxDets=100]	0.853	0.800	0.740
AR @[IoU=0.50:0.95 | area=small | maxDets=100]	−1.000	−1.000	−1.000
AR @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.800	0.700	0.775
AR @[IoU=0.50:0.95 | area=large | maxDets=100]	0.853	0.801	0.739

Table 4.2: Model Accuracy Results

predicted values. Using *metrics* module available in *scikit-learn* library, required accuracy metrics are obtained such as Average Precision, Average Recall and F1-Score. The comparisons can be seen in table 4.3.

The module provided two variants of metrics namely macro-averages and weighted averages. Macro-average gives equal importance to each gesture irrespective of its occurrence in the dataset, whereas weighted-average gives weightage to each class based on the number of samples that occurred in the dataset. As there is an imbalance in the number of samples per class, weighted-average metrics can explain the performance of the model better than simple traditional averages.

Model	**Avg.Precision**	**Avg.Recall**	**F1-Score**
SSD MobilenetV2	82.9	85.3	84.1
Faster R-CNN InceptionV2	75.5	79.9	77.6
R-FCN Resnet101	66.6	72.9	69.6
mpHands(weighted)	96.5	85.8	90.6
mpHands(macro)	84.6	74.9	79.3

Table 4.3: Accuracy Metrics Comparison

The timing analysis is done to understand how fast each model is processing a frame and producing the classification output. The experiment has been done on a system consisting of the GPU device, Tesla V100-PCIE-32GB. All the images in the test set are processed using a model and the total time lapsed is divided by the number of images. It can be observed from the results table 4.4 that mpHands model has outperformed all other models massively. The keypoint detection model is processing frames nearly 1.5 times faster than the next fastest model SSD. Faster R-CNN and R-FCN are very poor in processing speeds, which are nowhere near the real-time standards, i.e, 25FPS.

Model	**Frame Rate(FPS)**	**Inference Time(ms)**
SSD MobilenetV2	37.93	26.36
Faster R-CNN InceptionV2	2.31	432.9
R-FCN Resnet101	1.58	629.8
MediaPipe Hands	58.27	17.16

Table 4.4: Gesture Detection Timing Analysis

From the result obtained, it can be inferred that mpHands is performing better than the popular object detection models. When macro-averages are considered, the model is on par with the SSD model in terms of accuracy, but there is a clear edge in the case of weighted averages. The keypoint detection model has impressed more in the processing speeds by detecting gestures at nearly twice the speed of real-time requirements.

Chapter 5

Conclusions and Future Work

In this thesis work, we have verified the proof of concept for marker-less augmentation over the human hand. The proposed framework takes the help of the Mediapipe Hands model for hand tracking and feature detection. It uses the theory behind camera perspective transformation to project 3D objects in the real-world scene onto the image plane. The camera parameters are approximated by camera pose estimation and used to construct view and projection matrices for the 3D object. The framework uses hand gestures to interact with virtual objects such as registering, zooming, rotating around axes, etc. The Mediapipe Hands model is the best technique that can fulfil the marker-less feature of this framework as it performs better than the popular object detection models such as SSD, Faster-RCNN and R-FCN in terms of accuracy and processing speed. The mpHands model performs poorly over high contrast images, where the entire palm acquires a uniform dark colour. This framework can be adopted in a classroom setting for enriching the learning experience by developing good teaching content and adding new features.

Though the framework has established the proof of concept, there are many areas to improve upon to make the entire experience more engaging. Following are a few problems to work on in the future:

1. One of the challenges faced is to implement the scaling effect when we move the left hand toward and away from the camera. Ideally, the object should become bigger when brought close to t.0he camera and smaller when taken away from it. Our future aim is to bring in this display effect.

2. It is essential to explore the area of graphical objects to enrich the quality of virtual objects. In the future, we would like to introduce new virtual objects with better textures and new features such as building a scene by registering multiple objects simultaneously.

3. As this work's main objective is to improve the teaching methodology, we plan to compose lecture content with our framework by gathering a series of teaching modules such as explaining graphs, DNA structure, etc.

Bibliography

[1] Tara J Brigham. Reality check: basics of augmented, virtual, and mixed reality. *Medical reference services quarterly*, 36(2):171–178, 2017.

[2] Christian Moro, Zane Štromberga, Athanasios Raikos, and Allan Stirling. The effectiveness of virtual and augmented reality in health sciences and medical anatomy. *Anatomical sciences education*, 10(6):549–559, 2017.

[3] Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997.

[4] Xiao Li, Bo Xu, Yue Teng, Yi-tian Ren, and Zhu-min Hu. Comparative research of ar and vr technology based on user experience. In *2014 International Conference on Management Science & Engineering 21th Annual Conference Proceedings*, pages 1820–1827. IEEE, 2014.

[5] Jon Peddie. *Augmented reality: Where we will all live*. Springer, 2017.

[6] Greg Kipper and Joseph Rampolla. *Augmented Reality: an emerging technologies guide to AR*. Elsevier, 2012.

[7] Marybeth Green, Joy Hill Lea, and Cheryl Lisa McNair. Reality check: Augmented reality for school libraries. *Teacher Librarian*, 41(5):28, 2014.

[8] Antonia Cascales, Isabel Laguna, David Pérez-López, Pascual Perona, and Manuel Contero. An experience on natural sciences augmented reality contents for preschoolers. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 103–112. Springer, 2013.

[9] Patrick Maier and Gudrun Klinker. Augmented chemical reactions: An augmented reality tool to support chemistry teaching. In *2013 2nd Experiment@ International Conference (exp. at'13)*, pages 164–165. IEEE, 2013.

[10] Matt Bower, Cathie Howe, Nerida McCredie, Austin Robinson, and David Grover. Augmented reality in education–cases, places and potentials. *Educational Media International*, 51(1):1–15, 2014.

[11] Hong-Quan Le and Jee-In Kim. An augmented reality application with hand gestures for learning 3d geometry. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 34–41, 2017.

[12] Juhás Martin and Juhásová Bohuslava. Augmented reality as an instrument for teaching industrial automation. In *2018 Cybernetics Informatics (K I)*, pages 1–5, 2018.

[13] Roghayeh Barmaki, Kevin Yu, Rebecca Pearlman, Richard Shingles, Felix Bork, Greg M Osgood, and Nassir Navab. Enhancement of anatomical education using augmented reality: An empirical study of body painting. *Anatomical sciences education*, 12(6):599–609, 2019.

[14] Facebook. Oculus quest hand tracking. https://www.oculus.com.

[15] Snap Inc. Snapchat lens studio. https://lensstudio.snapchat.com.

[16] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[17] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. In *Computer Graphics Forum*, volume 34, pages 101–114. Wiley Online Library, 2015.

[18] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Self-supervised 3d hand pose estimation through training by fitting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10853–10862, 2019.

[19] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] Monica Gruosso, Nicola Capece, Ugo Erra, and Francesco Angiolillo. A preliminary investigation into a deep learning implementation for hand tracking on mobile devices. In *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 380–385. IEEE, 2020.

[21] Yangang Wang, Cong Peng, and Yebin Liu. Mask-pose cascaded cnn for 2d hand pose estimation from single color image. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(11):3258–3268, 2019.

[22] Byung-Kuk Seo, Junyeoung Choi, Jae-Hyek Han, Hanhoon Park, and Jong-Il Park. One-handed interaction with augmented virtual objects on mobile devices. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pages 1–6, 2008.

[23] Yuan Li, Xinggang Wang, Wenyu Liu, and Bin Feng. Pose anchor: A single-stage hand keypoint detection network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):2104–2113, 2020.

[24] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[25] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[26] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[29] R Girshick. Fast r-cnn. computer science. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[31] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.