ABSTRACT

The application is made possible by on-device and real-time machine learning algorithms that suss out the position of the shoes in space while accounting for color, texture, and lighting variations, plus a fully equipped model building software such as “Blender” is used to create 3D shoe models. The end result is foot-tracking tech that’s robust enough to adapt to different camera angles and follow footsteps as feet move and rotate.

TABLE OF CONTENTS

TABLE OF ABBREVIATIONS

3-D	Three Dimensional
APP	Application
AR	Augmented Reality
C#	C Sharp
HSV	Hue, Saturation, and Value
UI	User Interface
VS	Visual Studio
S/W	Software

CHAPTER 1

INTRODUCTION

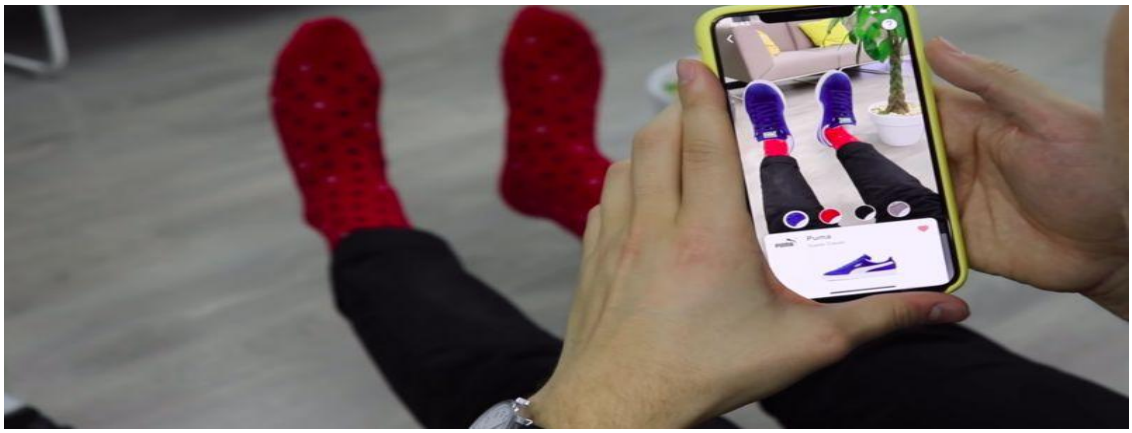
ABOUT TRY ON SHOE

Till date, online users were buying products from many e-commerce online Apps where to a certain level they were satisfied. But after the introduction of the AR technology online users can experience the new level of shopping experience.

Earlier we have to make a visit to your nearby local mall or shopping center, find a shoe store, and then you have to track someone down to bring out different sizes, different colors, different designs.

However, thanks to the AR tryon shoe basedon augmented reality technology and python, you can now try on multiple shoes, see how they look .

Likewise, a new idea of virtually experiencing the shoe using the smartphones took users to another level. Talking about this technology it is built so customers can digitally image (scan) their feet. The Aim of this try on app is to make accurate decisions when users are deciding what shoe and size to buy before they get it online.



CHAPTER 2
REQUIREMENT ANALYSIS

Requirement:

A S/W or application which can detect a foot and superimpose a 3D shoe model on it.

System Specifications:

8 GB RAM

i5 intel core processor

External GPU for good rendering

Software:

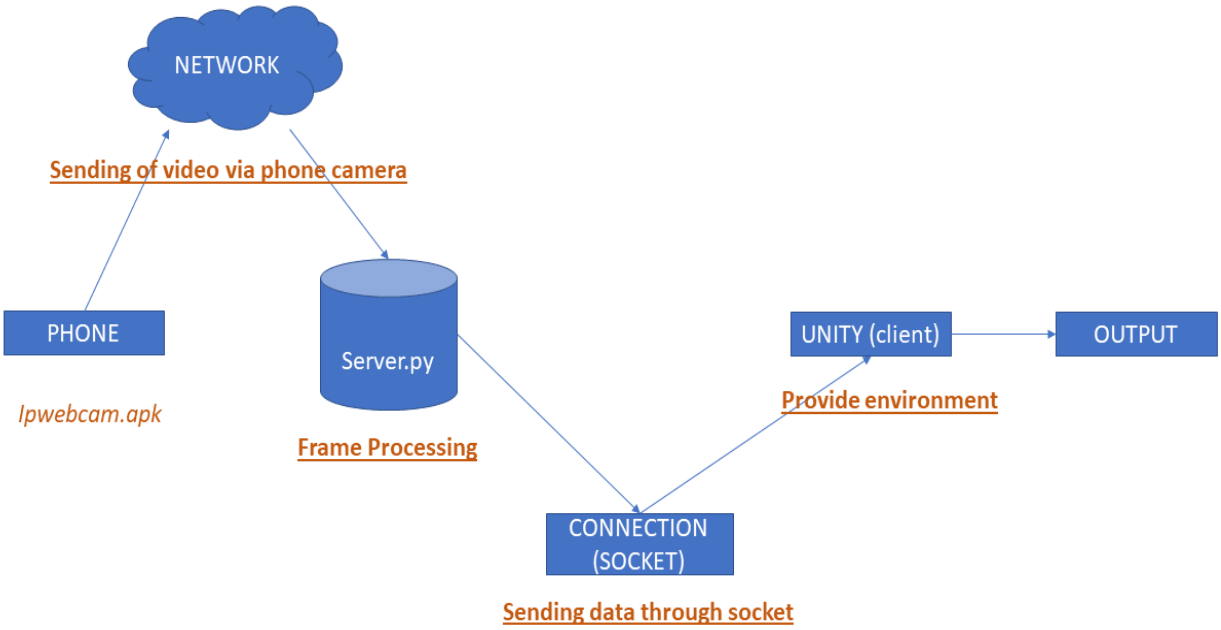
Unity

Blender

PyCharm

VS Code

DFD:


NETWORK
Sending of video via phone camera
PHONE
Ipwebcam.apk
Server.py
Frame Processing
UNITY (client)
OUTPUT
Provide environment
CONNECTION (SOCKET)
Sending data through socket

CHAPTER 3

SOFTWARE AND PROJECT DESIGN

As we wanted to create a try on virtual try on shoe application, we decided to create our own user interface for the same. During our research, we found that there were many existing pre-built snapchat templates available for performing the task. However we mutually decided to not let us take the easy route. Therefore we created the whole user interface using a combination of c# and python with integrated unity platform support.

Now we are going to recognize foot features from a video sequence. To recognize these features from a live video sequence, we first need to take out the foot region alone removing all the unwanted portions in the video sequence. After segmenting the foot region, we then send the video sequence of foot with the foot features to the unity game engine. On the basis of these features we placed a 3-D self created shoe model on the video sequence.

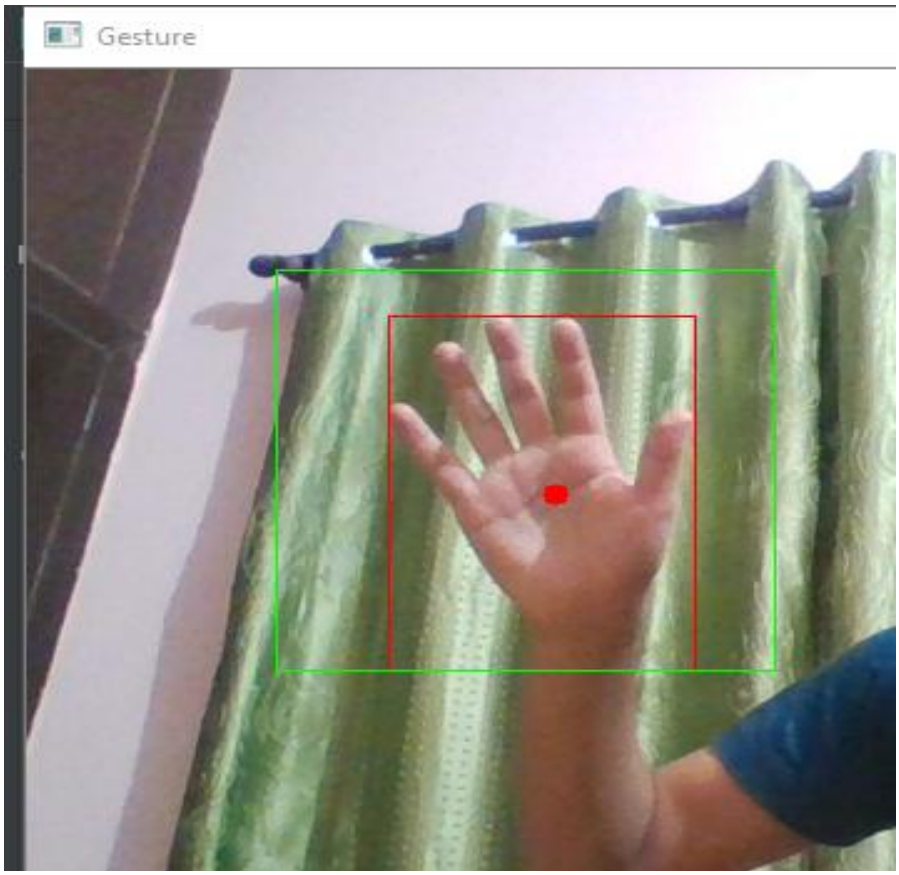
Steps followed:

1. Video capturing:

Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

Steps to capture a video:

- Use cv2.VideoCapture() to get a video capture object for the camera.
- Set up an infinite while loop and use the read() method to read the frames using the above created object.
- Use cv2.imshow() method to show the frames in the video.
- Breaks the loop when the user clicks a specific key.



2. Image segmentation:

The original video is captured through web cam of laptop and from the video each frame is used for segmentation. The background of these images is identical. So, it is easy and effective to detect the foot region from the original image using the background subtraction method. However, in some cases, there are other moving objects included in the result of background subtraction. The skin color can be used to discriminate the foot region from the other moving objects. The color of the skin is measured with the HSV model. The HSV (hue, saturation, and value) range value of the skin color is from 0, 48, 80, to 20, 255, 255 respectively. The image of the detected foot is resized to 640 X 480 to make the feature recognition invariant to image scale.

3. Counters detection:

The outline or the boundary of the object of interest. A contour is defined as the line that joins all the points along the boundary of an image that have the same intensity. cv2.findContours() function in OpenCV help us find contours in a binary image. If you pass a binary image to this function, it returns a list of all the contours in the image. Each of the elements in this list is a numpy array that represents the (x, y) coordinate of the boundary points of the contour (or the object).

4. ConvexHull:

Convex Hull will look similar to contour approximation, but it is not (Both may provide same results in some cases). Here, cv.convexHull() function checks a curve for convexity defects and corrects it. Generally speaking, convex curves are the curves which are always bulged out, or at-least flat. And if it is bulged inside, it is called convexity defects.

5. Max and Min Hull points:

Since we needed to calculate the length of foot so for that we calculated the max and min hull points on x and y axis from the convex hull points we got in the previous steps.

6. Calculating the mid points:

we need mid points to locate our to locate our foot in unity environment.To calculate mid points we used recursive mid point thermo. We first calculated the mid points of x axis using the min and max points of x axis and mid point of y axis using the max and min points of y axis.

7. Setting up connection:

We need to set a connection between python and unity so that they can communicate. Steps followed to make the connection:

- We used python as a server and unity as a client.
- Using socket programming we are connecting those two.
- We used TCP protocol

8. Sending data from Python:

After the connection was setup we are sending certain important data by converting them into byte stream.

Data sent:

- Byte array length
- Image
- Max min hull points
- Centroid

9. Receiving data from Python:

- After establishing the connection, we now receive the byte data inside unity , for this purpose we used socket connection using C# .As we click the play button the “start” function will execute within which we are setting the height and width for our canvas.
- We are then executing the function “init socket” ,within which we are creating threads and within the threads we are calling a function “socket receive”.
- Within “socket receive” there is another function called “socket connect” which we are using to establish connection with the help of TCP/IP protocol.
- Now after this we are receiving byte data within the port and also its length, if the length is zero then we will reconnect to the server.
- We then convert the received length into string first and then integer.
- We are now receiving the image through socket, alongside hull points.
- For each frame(thread) the same process will be repeated.

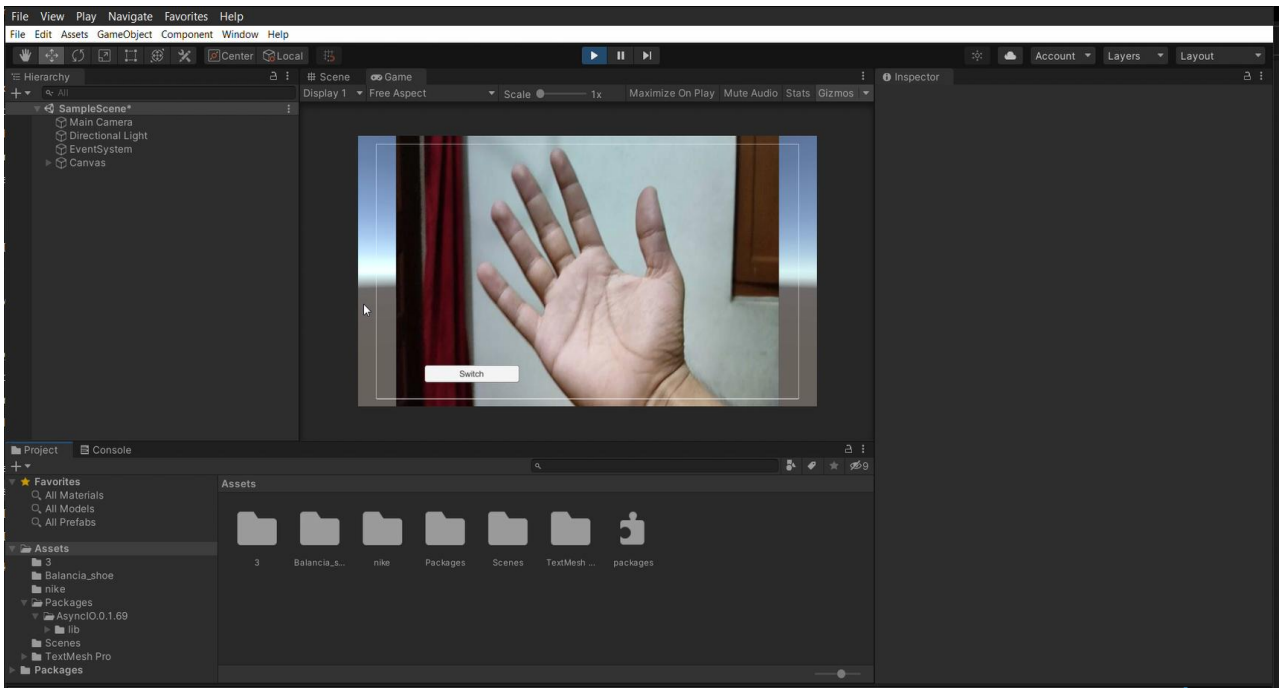
10. Placing Image in Unity

To place the image in unity we use the canvas game object.

The Canvas is the area that all UI elements should be inside. The Canvas is a Game Object with a Canvas component on it, and all UI elements must be children of such a Canvas.

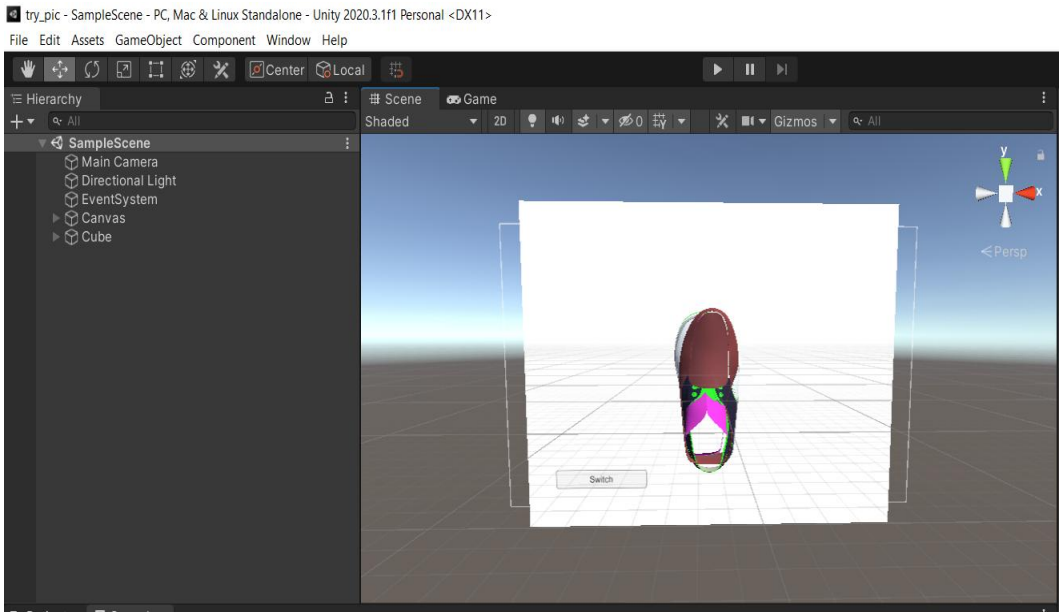
The Canvas area is shown as a rectangle in the Scene View. This makes it easy to position UI elements without needing to have the Game View visible at all times.

Canvas uses the EventSystem object to help the Messaging System.



11. Creating and Placing models

- We created the 3D Shoe Models using Blender .
- Next, we created a repository within “assets” consisting of all the shoe models.
- The shoe models to be displayed were then placed on the canvas overlapping one another.



12. Switching between the models

- For switching between the models we created a additional C# script.
- Within that we created multiple game objects and switched between the shoe models using Switch Avatar Method().
- The multiple game objects were then the given the reference of respective shoe models.
- We also designed a switch Button to switch among the models.

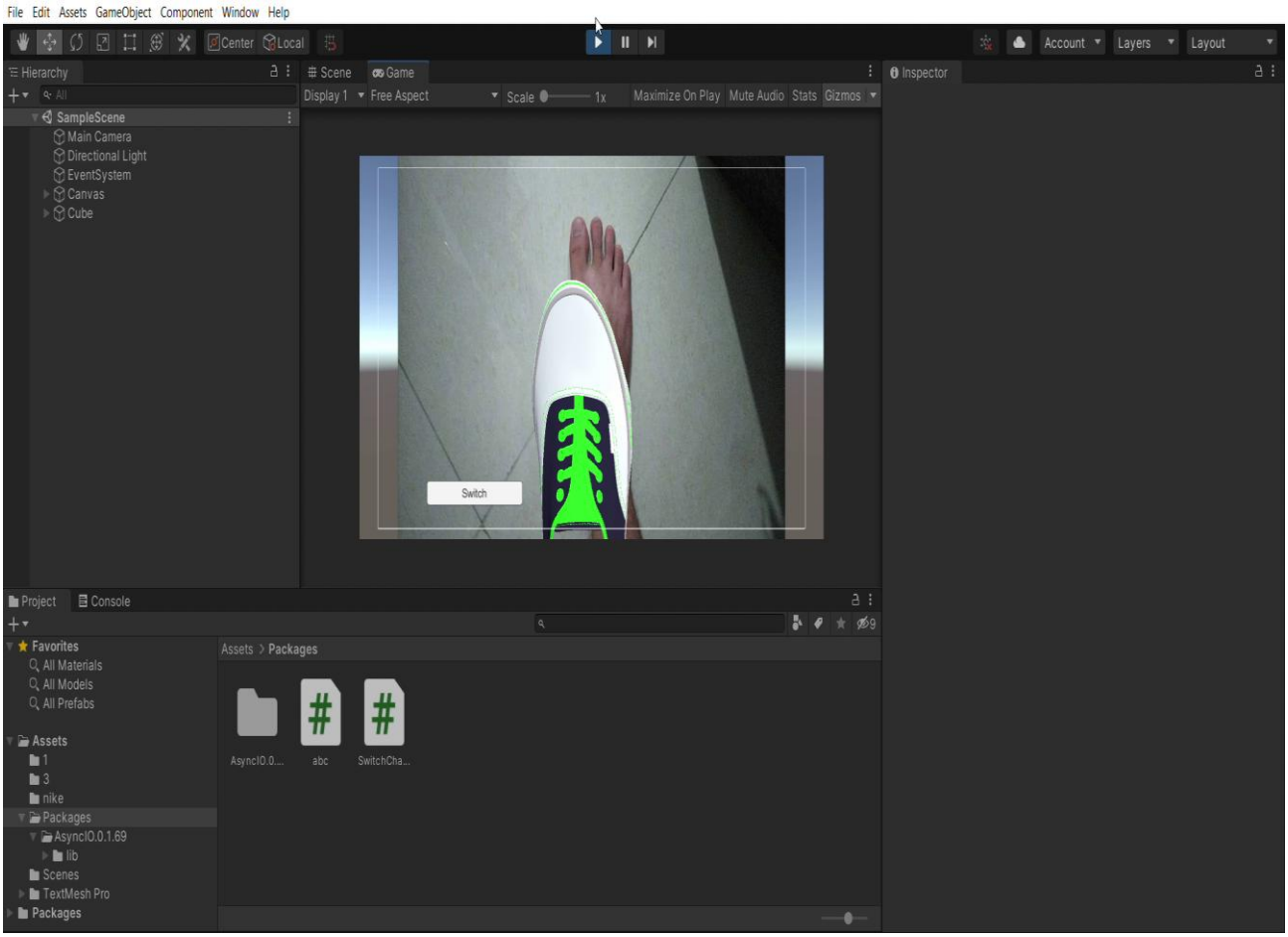
13. Playing the application

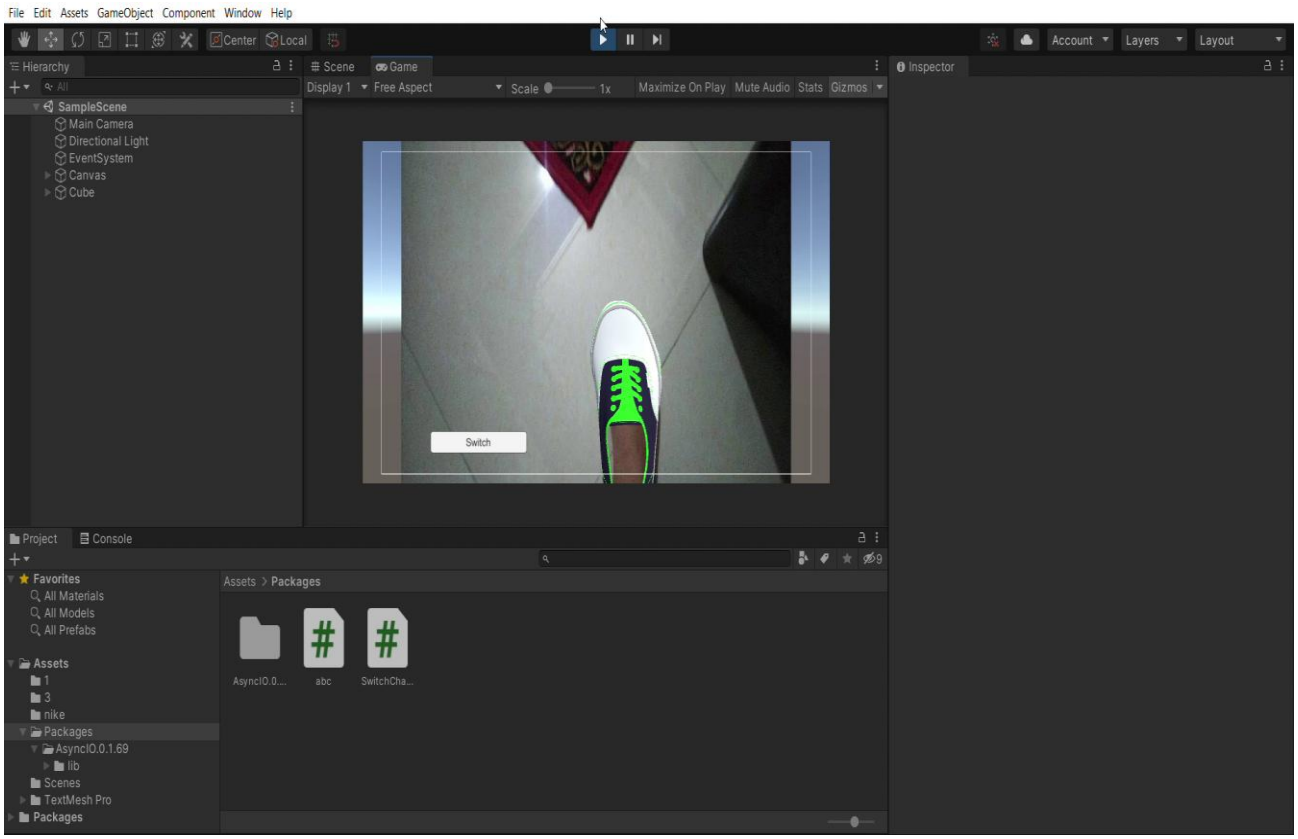
- Finally we place the 3 D model on top of our feet using the hull points provided.
- We also used “localpos()” for repositioning the model and “localscale()” for rescaling the model.
- Now we can also switch amongst the models using the button.

CHAPTER 4

RESULT/TESTING OF PROJECT / SOFTWARE

TESTING RESULTS:




File Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Hierarchy
SampleScene
Main Camera
Directional Light
EventSystem
Canvas
Cube
Scene Game
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Switch
Inspector
Project Console
Favorites
All Materials
All Models
All Prefabs
Assets
1
3
nike
Packages
AsyncIO.0.1.69
lib
Scenes
TextMesh Pro
Packages
Assets > Packages
AsyncIO.0....
abc
SwitchCha...

Chapter 5
Conclusion and Future Scope

Finally ,this project utilizes usage of one of the most popular modern technologies that is augmented reality with python support.

Augmented reality has been a hot topic in software development circles for a number of years, but it's getting renewed focus and attention. Augmented reality is a technology that works on computer vision based recognition algorithms to augment sound, video, graphics and other sensor based inputs on real world objects using the camera of your device. It is a good way to render real world information and present it in an interactive way so that virtual elements become part of the real world.

A simple augmented reality use case is: a user captures the image of a real-world object, and the underlying platform detects a marker, which triggers it to add a virtual object on top of the real-world image and displays on your camera screen.

The application that we have created utilises a new technology like augmented reality and gives it a commercial aspect as well. Many brands like Gucci, Reebok,etc have effectively used it to promote their products by giving the user a real time experience of trying their dream shoes virtually.

The user can try out as many models as he likes without going out to the showrooms of these brands. They can simply lay on their bed and observe how a given shoe model looks on their feet. This has definitely been a blessing to these brands in marketing terms as well and helped them harness more revenues for their products. Our application also offers a similar future prospective growth and can be used to promote shoe brands and can definitely prove substantial in promoting market growth of the clients.

APPENDIX A

Python code:

```
import cv2
import socket
import time
import cv2 as cv
import numpyas np
import requests
import imutils

url = "http://192.168.43.29:8080/shot.jpg"

defcentroid(a,b,c,d):
len_mid_y = (a[1] + b[1])/2
if abs(a[0]) <abs(b[0]):
len_mid_x = a[0]
else:
len_mid_x = b[0]

wid_mid_y = (c[0]+d[0])/2
wid_mid_x = (c[1]+d[1])/2


cen_mid_x = (len_mid_x + wid_mid_x)/2
cen_mid_y = (len_mid_y + wid_mid_y)/2

return [cen_mid_x,cen_mid_y]

itr = 1
while(True):

    HOST = '127.0.0.1'
PORT = 9000

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)     # tcp
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)   # reuse tcp
sock.bind((HOST, PORT))
sock.listen(1)

print('Wait for connection...')
    (client, adr) = sock.accept()
print("Client Info: ", client, adr)

frame_rate = 0.8
prev = 0
fps_time = 1.0
while(True):
img_resp = requests.get(url)
img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
img = cv2.imdecode(img_arr, -1)
img = imutils.resize(img, width=640, height=480)
        img2 = imutils.resize(img, width=640, height=480)
```

```
        cv2.imshow("Android_cam", img)
        frame = img
        frame2 = img2

hsvim = cv.cvtColor(frame, cv.COLOR_BGR2HSV)
        lower = np.array([0, 48, 80], dtype= "uint8")
        upper = np.array([20, 255, 255], dtype= "uint8")
skinRegionHSV = cv.inRange(hsvim, lower, upper)
        blurred = cv.blur(skinRegionHSV, (2,2))
rt,thresh = cv.threshold(blurred,0,255,cv.THRESH_BINARY)

        contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
        contours = max(contours, key=lambda x: cv.contourArea(x))

cv.drawContours(frame, [contours], -1, (255,255,0), 2)

        hull = cv.convexHull(contours)
cv.drawContours(frame, [hull], -1, (0, 255, 255), 2)

x_axis = []
y_axis = []

for iin hull:
x_axis.append(i[0][0])
y_axis.append(i[0][1])

x_max, x_min = max(x_axis), min(x_axis)
y_max, y_min = max(y_axis), min(y_axis)


sent_arr = []
sent_arr.append(x_max)
sent_arr.append(x_min)
sent_arr.append(y_max)
sent_arr.append(y_min)

centroid_val = []
y_m_flag = 0
x_m_flag = 0
y_flag = 0
x_flag = 0
for h in hull:
            x = h[0][0]
            y = h[0][1]
if y_m_flag == 0 and y == y_max:
centroid_val.append([x,y])
y_m_flag = 1
if y_flag == 0 and  y == y_min:
centroid_val.append([x,y])
y_flag = 1
if x_m_flag == 0 and x == x_max:
centroid_val.append([x,y])
x_m_flag = 1
if x_flag == 0 and x == x_min:
centroid_val.append([x,y])
x_flag = 1

cent = centroid(centroid_val[0], centroid_val[1], centroid_val[2],
centroid_val[3])
```

```
        result, encimg = cv2.imencode('.jpeg', frame2,
[int(cv2.IMWRITE_JPEG_QUALITY), 70])

        temp = client.recv(1024)


client.send(str(len(encimg.flatten())).encode('utf-8'))
print('number of frame :',itr)
itr+=1

temp = client.recv(1024)
client.send(encimg)
        temp = client.recv(1024)

keypoint_str = str(int(sent_arr[0])) + ',' + str(int(sent_arr[1])) + ',' +
str(int(sent_arr[2])) + ',' + str(int(sent_arr[3]))
client.send(bytes(keypoint_str, 'ascii'))

cent_str = keypoint_str = str(cent[0]) + ',' + str(cent[1])
client.send(bytes(cent_str, 'ascii'))

        cv2.putText(frame2, "FPS: %f" % (1.0/(time.time() - fps_time)),
(30, 30),  cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow('frame', frame2)
        cv2.waitKey(1)
fps_time = time.time()
time.sleep(0.3)
```

C# code:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

public class abc : MonoBehaviour
{
    string recvStr;
intflag;
public RawImagemy_img;
Socket serverSocket;
IPAddressip;
IPEndPointipEnd;
string[] arr = new string[1024];
string[] cent_arr = new string[1024];
byte[] recvData = new byte[1024];
byte[] sendData = new byte[1024];
byte[] centr = new byte[1024];
string cent;
byte[] recv_img;
intrecvLen;
byte[] recv_keypoint = new byte[1024];
string recv_keypoint_str;
public float shoe_scale = 2f;
public GameObject Cube;
float R_center_x = 0f;
float R_center_y = 0f;
intitr = 1;
Thread connectThread;

void InitSocket()
{

ip = IPAddress.Parse("127.0.0.1");
ipEnd = new IPEndPoint(ip, 9000);



connectThread = new Thread(new ThreadStart(SocketReceive));
connectThread.Start();
}

void SocketConnet()
{
if (serverSocket != null)
serverSocket.Close();
serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
print("ready to connect");
serverSocket.Connect(ipEnd);


}
```

```
void SocketSend(string sendStr)
{

sendData = new byte[1024];
sendData = Encoding.ASCII.GetBytes(sendStr);
serverSocket.Send(sendData, sendData.Length, SocketFlags.None);
}

void SocketReceive()
{
SocketConnet();

while (true)
    {
SocketSend("OK_1");

recvData = new byte[1024];
recvLen = serverSocket.Receive(recvData);
if (recvLen == 0)
    {
    print("reconnecting...");
SocketConnet();
continue;
    }

SocketSend("OK_2");

recvStr = Encoding.ASCII.GetString(recvData, 0, recvLen);
intimg_size = Convert.ToInt32(recvStr);
    //intcou = 0;
    flag = 0;
recv_img = new byte[img_size];
intrecv_img_len = serverSocket.Receive(recv_img);
if (img_size != recv_img_len)
    {
print("error");
}
else
{
flag = 1;

SocketSend("OK_3");

recv_keypoint = new byte[1024];
intrecv_keypoint_len = serverSocket.Receive(recv_keypoint);
recv_keypoint_str = Encoding.ASCII.GetString(recv_keypoint, 0,
recv_keypoint_len);

arr = recv_keypoint_str.Split(',');

centr = new byte[1024];
intrecv_cent = serverSocket.Receive(centr);
cent = Encoding.ASCII.GetString(centr, 0, recv_cent);
cent_arr = cent.Split(',');

}
}
}

void SocketQuit()
```

```
{

if (connectThread != null)
{
connectThread.Interrupt();
connectThread.Abort();
}

if (serverSocket != null)
serverSocket.Close();
print("diconnect");
}
void Start()
{
my_img.texture = new Texture2D(640, 480, TextureFormat.RGB24, false);
InitSocket();

}




void Update()
{


var texture = my_img.textureas Texture2D;


texture.LoadImage(recv_img);
print("load image" + itr++);
texture.Apply();
float f_len = Int32.Parse(arr[2]) - Int32.Parse(arr[3]);
float f_wid = Int32.Parse(arr[0]) - Int32.Parse(arr[1]);
R_center_y = (Int32.Parse(arr[2]) + Int32.Parse(arr[3])) / 2f;
R_center_x = (Int32.Parse(arr[0]) + Int32.Parse(arr[1])) / 2f;
float x = float.Parse(cent_arr[0]);
float y = float.Parse(cent_arr[1]);
Cube.GetComponent<Transform>().localPosition = new Vector3(x - 300f, -(y -
250f), -3);
print(f_wid + " " + f_len);
Cube.transform.localScale = new Vector3(f_wid / 90, f_len / 150, f_wid /
200);

}

void OnApplicationQuit()
{
SocketQuit();
}
}
```

References

1. https://docs.unity3d.com/Manual/ScriptingSection.html
2. https://docs.python.org/3/
3. https://docs.opencv.org/master/index.html