VISION BASED POSITION TRACKING FOR VIRTUAL REALITY

COURSE PROJECT

AM5011: VIRTUAL REALITY ENGINEERING

GUIDE : Dr. M MANIVANNAN

SUBMITTED BY :

RIDHI PUPPALA - ME15B133

DASARI ANANYANANDA - ME15B163

Introduction

Virtual reality is an extremely fast growing field due to its wide and varied applications in many areas of science and technology. Most of the mobile Headsets for Virtual Reality devices like Google Cardboard, Samsung gear VR and several other small company products currently lack a good position tracking system while they accommodate a head orientation tracking mechanism. They can track the rotation of the head using IMU sensors in the phone, but determining head position is difficult. This affects the overall immersive experience of these VR devices.

State of art VR Head Mounted Displays like HTC VIVE and Oculus use a set of IR LEDs and multiple high resolution cameras for tracking the position of the user in a constrained physical domain. These products are very expensive and not affordable to everyone. With VR becoming more popular day by day, there is need to make it affordable while not compromising on the overall immersive experience.

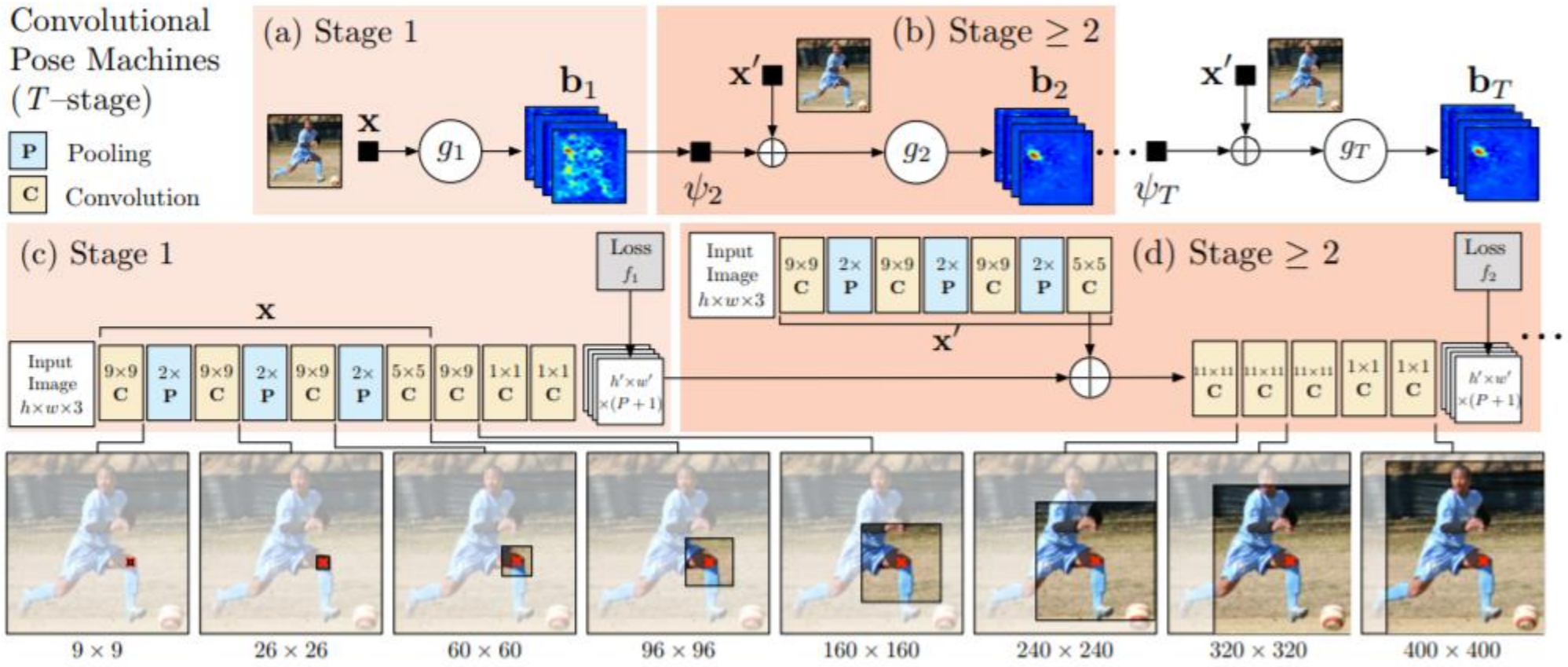
PROBLEM DEFINITION

Create an efficient and cost-effective position tracking system for smartphone virtual reality, to enhance user experience such that it can work in combination with head orientation tracking and virtual environment generation software. Aim is to develop a compact and modular setup for positional tracking in constrained indoor locations.

Literature Survey for Possible Approaches

- VSLAM
- PnP
- Monocular Tracking – VNECT
- Outside-in CNN based tracking (Final Approach)

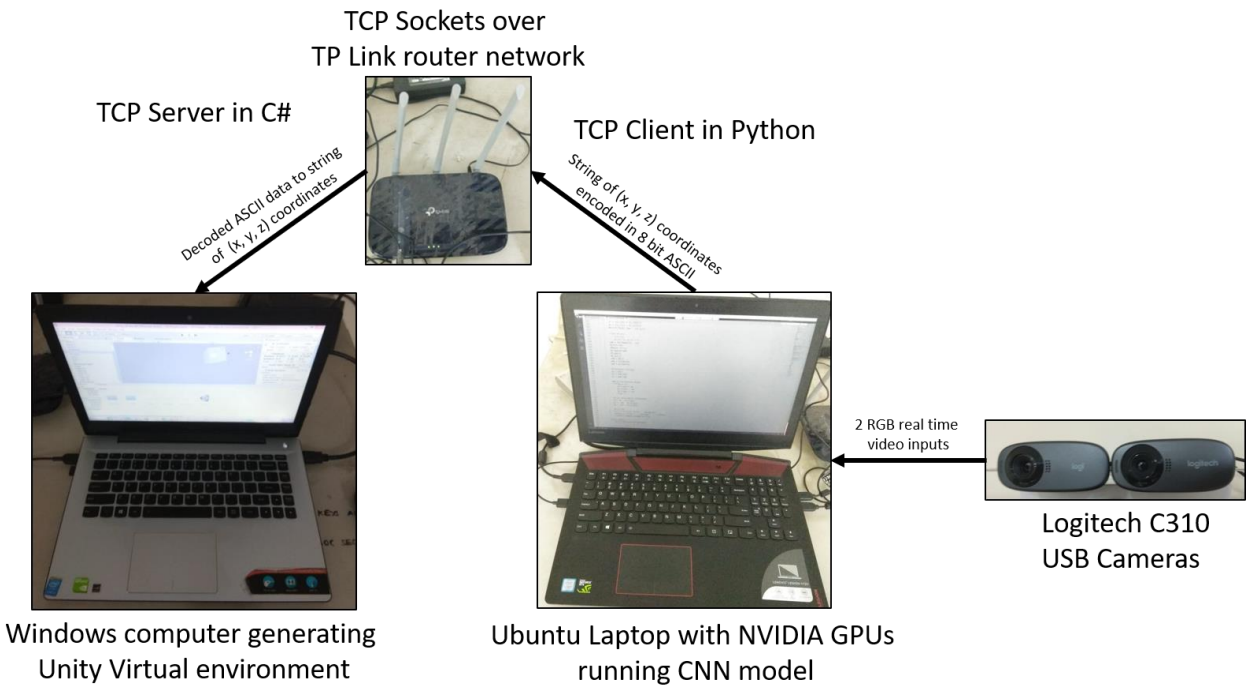
CNN Architecture

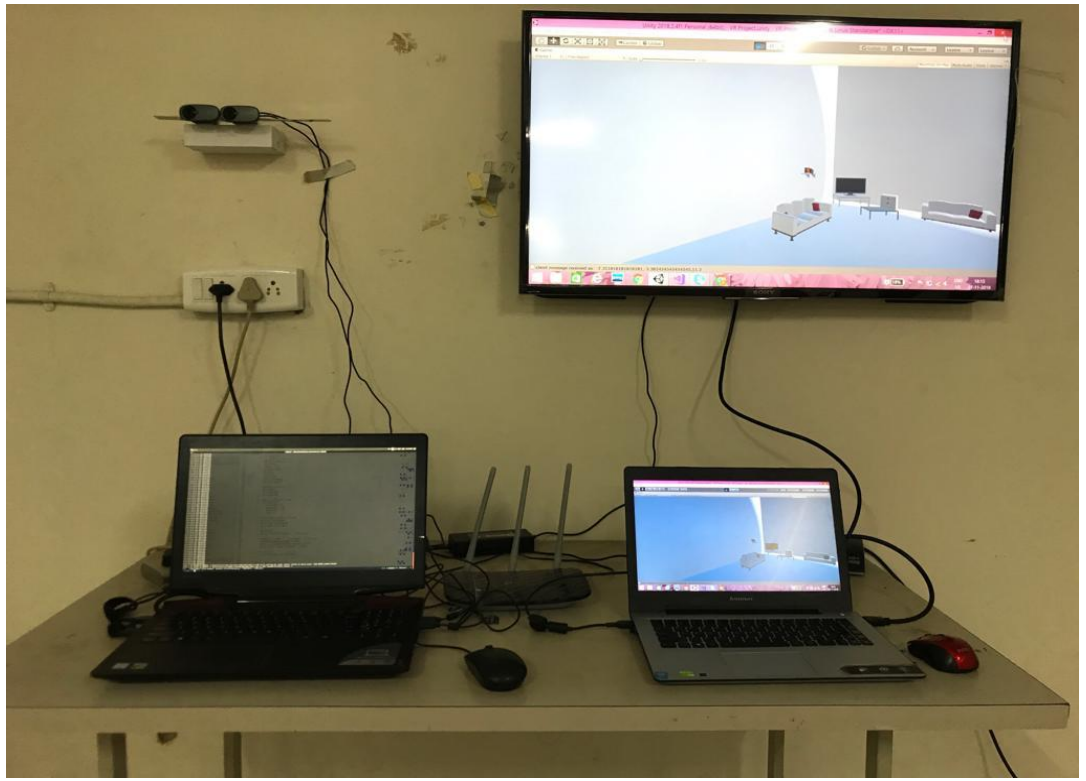


Hardware Specification

- Ubuntu 16.04 laptop with NVIDIA GPU to run the python CNN tracking code
- Windows laptop to generate and update Unity virtual environment
- TP Link Wifi router to establish communication between the two laptops over TCP based Sockets
- Two Logitech C310 USB Cameras with 4.4mm focal length, 720p resolution and 60° Field of view

Architecture





3D world coordinate estimation

$$z = f^{*}b / (xl - xr) = f^{*}b/d$$

$$x = xl^{*}z/f = b + xr^{*}z/f$$

$$y = yl^{*}z/f = yr^{*}z/f$$

Here,

f = focal length in mm

b = baseline distance between two cameras (lens to lens center distance)

(xl, yl) = X and Y pixel coordinates of feature from left camera

(xr, yr) = X and Y pixel coordinates of feature from right camera

(x,y,z) = 3D world coordinates

d = disparity between two images

TCP Socket Programming

```
private void ListenForIncommingRequests()
{
    try
    {
        // Create listener on localhost port 8052.
        tcpListener = new TcpListener(IPAddress.Parse("192.168.1.109" + ""), 52595);
        tcpListener.Start();
        Debug.Log("Server is listening");
        Byte[] bytes = new Byte[1024];
        while (true)
        {
            using (connectedTcpClient = tcpListener.AcceptTcpClient())
            {
                // Get a stream object for reading
                using (NetworkStream stream = connectedTcpClient.GetStream())
                {
                    int length;
                    // Read incomming stream into byte arrary.
                    while ((length = stream.Read(bytes, 0, bytes.Length)) != 0)
                    {
                        var incommingData = new byte[length];
                        Array.Copy(bytes, 0, incommingData, 0, length);
                        // Convert byte array to string message.
                        clientMessage = Encoding.ASCII.GetString(incommingData);
                        Debug.Log("client message received as: " + clientMessage);
                    }
                }
            }
        }
```

Code snippet of C# Server

```
import socket
HOST = '192.168.1.109' #SERVER IP
PORT = 52595
BUFFERSIZE = 1024
# TCP Socket client
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((HOST,PORT))
print("Socket Connection Established")
while True:
    s.sendall(campose.encode("utf-8"))
s.close()
```

Code snippet of Python Client

TCP Socket Programming

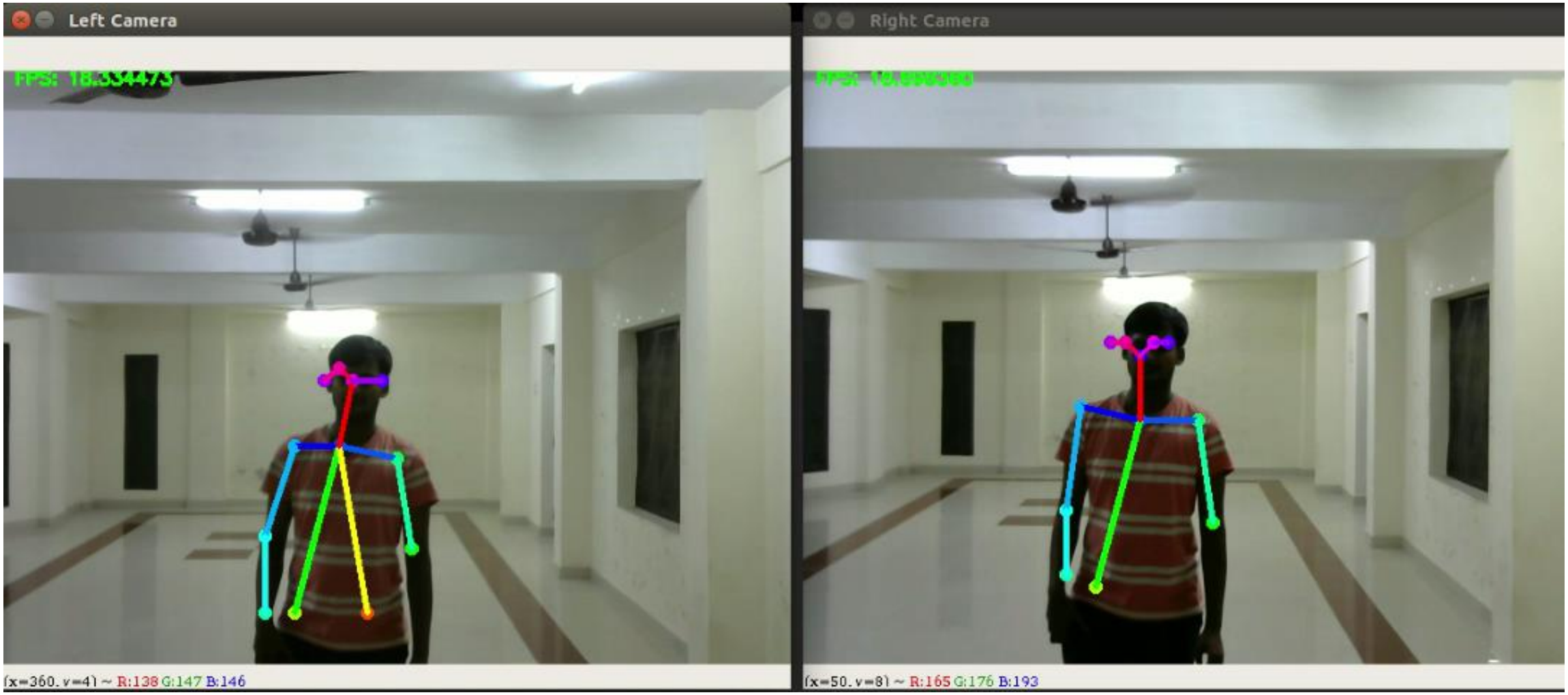
```
void Update()
{
    time += Time.deltaTime;

    if (time >= interpolationPeriod)
    {
        time = time - interpolationPeriod;
    }
    string[] data = clientMessage.Split(',');
    Vector3 pose = new Vector3(Single.Parse(data[0]), Single.Parse(data[1]), Single.Parse(data[2]));
    Cam.transform.position = pose;
    Cam.transform.rotation = Quaternion.Euler(0.0f, 45.0f, 0.0f);

}
```

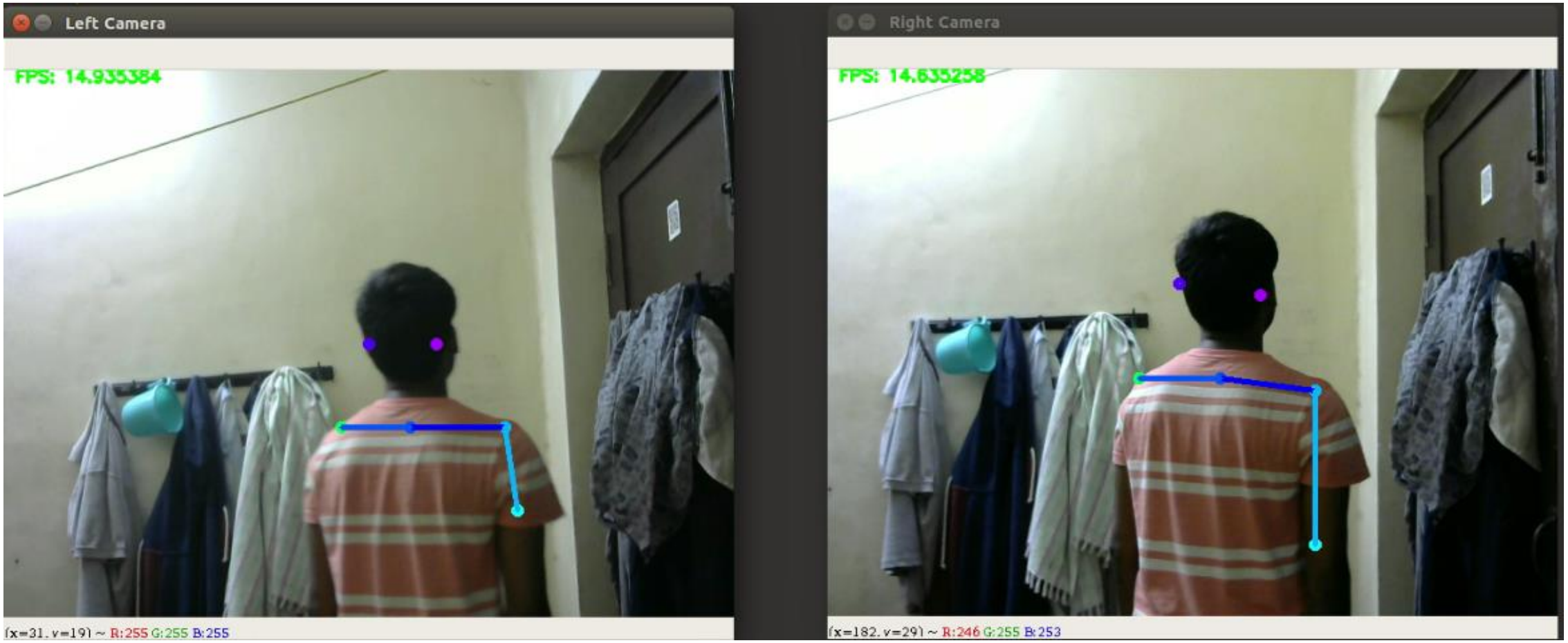
C# script showing update function

Results



Joint tracking for left and right cameras implemented on a Ubuntu laptop with NVIDIA GPUs

Results



Shoulder tracking performance with the human not facing the camera justifies choice of shoulder joint for position tracking of human body for VR

Results

Shoulder tracking performance even with the eyes occluded by white box which can be assumed to be a Head mounted display (HMD)

CHALLENGES FACED

- Computationally expensive CNN models were used which demanded for high computational resources/processors and GPUs
- Precise mounting of camera is essential for accurate estimation of world coordinates, any lateral and transverse offset between camera centers can have drastic effect on depth estimation (z-coordinate) and tracking accuracy
- Noise in CNN based shoulder joint location estimates in both cameras also has an adverse effect on tracking accuracy
- Fluctuations in CNN based shoulder joint location estimates in both cameras led to fluctuations in the Unity virtual environments which was later eliminated by Median filtering technique
- Issue in interfacing TCP Sockets between Ubuntu and C# computers and parsing ASCII encoded data in Unity which caused a lag in unity environment updates
- VR tracking system was lagging in case of rapid movements

References

[1] https://github.com/CMU-Perceptual-Computing-Lab/openpose
[2] http://gvv.mpi-inf.mpg.de/projects/VNect/
[3] https://courses.cs.washington.edu/courses/cse576/book/ch12.pdf
[4]https://gist.github.com/danielbierwirth/0636650b005834204cb19ef5ae6c cedb
[5] https://nptel.ac.in/courses/106106138/
[6]http://www.vudream.com/slam-inside-out-positional-tracking-solution-for-vr-and-ar/
[7]https://dspace.mit.edu/bitstream/handle/1721.1/36832/16-412JSpring2004/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring2004/A3C5517F-C092-4554-AA43-232DC74609B3/0/1Aslam_blas_report.pdf
[8] Eric Marchand, Hideaki Uchiyama, Fabien Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. IEEE Transactions on Visualization and Computer Graphics, Institute of Electrical and Electronics Engineers, 2016, 22 (12), pp.2633 - 2651.

Thank You