Technische Universität Berlin

Department of Telecommunication Systems
Open Distributed Systems

Faculty IV -
Electrical Engineering and Computer Science
https://www.ods.tu-berlin.de/

Master's Thesis
as a part of the Masters in Computer Science Program

Object Detection with use-case using WebXR

Shaon Debnath

Matriculation Number: 380601
04.07.2019

Supervised by

Prof. Dr. Manfred Hauswirth
Prof. Dr.-Ing. Ina Schieferdecker
Dipl.-Ing. Louay Bassbouss

FOKUS Institute
Kaiserin-Augusta-Allee 31
10589 Berlin

This dissertation originated in cooperation with the Fraunhofer Institute for Open Communication Systems (FOKUS).

First of all I would like to thank Prof. Dr. Manfred Hauswirth and Prof. Dr.-Ing. Ina Schieferdecke at the Fraunhofer Institute FOKUS for giving me the opportunity to carry out state of the art research in this field.

Furthermore, I would like to thank Dipl.-Ing. Louay Bassbouss for creating an opportunity to write my thesis in this arena and for guiding me in this thesis.

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

Berlin, 04.07.2019

. . . . . .

(Shaon Debnath)

Abstract

The concept of web-based cross reality (webXR) which includes virtual reality (VR), augmented reality (AR) and mixed reality (MR) has emerged in a big scale recently, but the supported user interaction devices are limited. In virtual reality, an artificial environment is created through software in such a way that the user believes it as a real environment. Augmented reality gives a live view of physical and real-world environment where the elements are augmented by computer-generated information. WebXR-device-API (also called WebXR-API) extends the concepts of native WebVR implementation by incorporating the AR capabilities within them. The name "WebXR apps" explains the concept itself, i.e. the XR applications which are able to run on the web browsers. WebXR-API allows developers to access virtual reality (VR) and augmented reality (AR) supported devices, including sensors and head-mounted displays to develop XR apps for the Web. That means, users will be able to run XR applications on the web browsers without installing the software on their respective devices. In this thesis, we have discussed on the key technologies behind the XR application, the frameworks and APIs to develop XR application for the web distribution. We have focused on different real-time object detection frameworks and their performance in webXR mobile app, controlling other devices through webXR app, implementation of XR app inside a regular web-page, 3D model visualization, and 3D XR game for web. We have considered the different use-cases, where XR application through web distribution is more appropriate than app store model. In the use-cases, we have chosen different sectors such as e-commerce, education and game arena. To check the feasibility of each use-case we have developed six different applications. We have described the implementation of these applications to give a demonstration on how webXR apps can be developed and different libraries can be integrated with webXR-API. We have also discussed their performance based on different matrices, and the challenges we faced while working with WebXR-API.

Keywords: WebXR-Device-API, WebXR apps, Augmented reality, Virtual reality, Mixed Reality, Object Detection, 3D model Visualization, Web-based XR Games

Zusammenfassung

Das Konzept der webbasierten Cross Reality (webXR), welches Virtual Reality (VR), Augmented Reality (AR) und Mixed Reality (MR) umfasst, findet in letzter mehr und mehr Verbreitung, aber die unterstützten Benutzerschnittstellen sind begrenzt. In der virtuellen Realität wird durch Software eine künstliche Umgebung so geschaffen, dass der Benutzer sie als reale Umgebung betrachtet. Augmented Reality gewährt einen Blick auf die physische und reale Umgebung, in der Elemente durch computergenerierte Informationen ergänzt werden. WebXR-device-API (auch WebXR-API genannt) erweitert die Konzepte der nativen WebVR-Implementierung, indem sie die AR-Funktionen integriert. Der Name "WebXR-Apps" erklärt das Konzept selbst, d.h. XR-Applikationen die auf Webbrowsern laufen können. WebXR-API ermöglicht Entwicklern den Zugriff auf Virtual Reality (VR) und Augmented Reality (AR) unterstützte Geräte, einschließlich Sensoren und Head Mounted Displays, um XR-Apps für das Web zu entwickeln. Das bedeutet,Benutzer können XR-Anwendungen auf Webbrowsern ausführen, ohne die Software auf ihren jeweiligen Geräten zu installieren. In dieser Arbeit haben wir über die Schlüsseltechnologien hinter der XR-Anwendung, die Frameworks und APIs diskutiert, um XR-Anwendungen für die Web-Distribution zu entwickeln. Wir haben uns auf verschiedene Echtzeit-Objekterkennungs-Frameworks und deren Leistung in der mobilen webXR-Anwendung konzentriert, auf die Steuerung anderer Geräte über die webXR-Anwendung, die Implementierung der XR-Anwendung innerhalb einer regulären Webseite, auf 3D-Modellvisualisierung und auf 3D-XR-Spiele für das Web. Wir haben die verschiedenen Anwendungsfälle berücksichtigt, in denen XR-Anwendungen durch Web-Distribution besser geeignet sind als App Store Modelle. In den Anwendungsfällen haben wir verschiedene Sektoren wie E-Commerce, Bildung und Spiele ausgewählt. Um die Machbarkeit der einzelnen Anwendungsfälle zu überprüfen, haben wir sechs verschiedene Anwendungen entwickelt. Wir haben die Implementierung dieser Anwendungen beschrieben, um zu demonstrieren, wie webXR-Anwendungen entwickelt und verschiedene Bibliotheken mit webXR-API integriert werden können. Wir haben auch ihre Leistung auf der Grundlage verschiedener Matrizen besprochen und die Herausforderungen, welche sich uns bei der Arbeit mit WebXR-API gestellt haben.

Schlüsselwörter: WebXR-Device-API, WebXR apps, Augmented Reality, Virtuelle Realität, Mixed Reality, Objekterkennung, 3D-Modellvisualisierung, Web-basierte XR-Spiele

Contents

List of Figures

1 Introduction

1.1 Motivation

We are experiencing now the growth of augmented reality (AR) platforms, devices, and frameworks at a massive scale. Platforms such as ARKit (for iOS) and ARCore (for Android) provide AR features on smartphones with standard camera flattening the need of motion cameras, which were earlier required for this kind of applications [Spe18]. Many applications with augmented reality (AR), virtual reality (VR) are released in the app store. The apps with AR and VR are getting popularity in different sectors such as 360 video, AR video or photo apps, e-commerce, game arena etc.

But there are some challenges for both AR and VR. Both need some dedicated hardware for the applications. They also need some supporting software to be installed and download the applications or setting up the sensors. Moreover, developers need to develop applications separately for iOS, Android or any other platforms. And also consumers need to download and install every application separately. There are many use-cases which does not suit with app store model, rather suits with web distribution model. To overcome these limitations of VR and AR, WebXR-API provides developers the ability to create a wide range of virtual reality (VR) and augmented reality (AR) experiences for all type of devices. [Wil17].

WebXR-device-API (also known as WebXR-API) provides support for accessing VR and AR devices including sensors and head-mounted displays on the Web. WebXR-API allows to control the render of reality inside the browser. It also provides support for many AR pages simultaneously. Using WebXR-API we can develop webXR apps easily targeting millions of devices with a single codebase [Moz18].

1.2 Problem Statement

Like other XR capable devices, smartphones are now releasing with powerful processors, cameras and graphics card which are capable to provide hardware support for XR applications. Lots of native XR applications can be found in the app store. But there are many use cases for XR applications, which do not fit in the app store model rather they are more appropriate for web distribution. Therefore, we need to consider XR application running on the top of browsers. But there are some problems and limitation to run XR application on the browser.

- Object detection and recognition is one of the important parts in Augmented reality. There are few challenges to implement object detection framework in mobile especially when the application runs through a web browser. The images captured through the mobile camera are not distortion free. These distortions may increase false negative result in object detection. Object recognition algorithms need a significant amount of computer resources and processing power. In real-time object detection application, in each frame object detection algorithm needs to be executed. Usually, apps run with 60 fps (frames per second). Hence it requires lots of processing power. Though the processing power of mobile devices is increasing nowadays but depending on the power of mobile, the same application may react differently. Therefore, during implementing object detection framework in an application, developers should develop it in such a way, so that the application does not dominate all the resource of device.

  Besides this, in terms of native application, applications receive some benefits as they are developed using native programming languages, and installed top of the operating system of the device. Therefore, application can directly access the hardware information. But for web application, it runs on the top of a web browser, therefore depending on the browser capabilities, performance may differ.

- Object visualization through web application can bring a great AR experience for the consumers. It is not feasible that every e-commerce site will be released as an application in app store to provide AR experience to the consumers. If we consider from the consumer's side, consumers also will not be interested to download all the apps for each website/seller. It is more relevant that consumers will search products on web, will go through different e-commerce web sites, and will experience augmented reality run-time from the website, without installing or downloading other software.

- Smartphone brought a revolution in the gaming industry. Beside game distribution through the app store, web-based games are also gaining much popularity. Developers are becoming interested, as they can target millions of devices with different operating systems using same base code. Facebook games [Fac19] are very popular for web-based gaming platform nowadays. Recently Google has announced their new cloud-based gaming platform Google Stadia[Goo19]. The stadia games can be played on google chrome browser or any supported browsers without having high graphics card or powerful processor from any devices i.e., laptop, tablet or mobile. To play stadia game, it just requires a good internet connection.

  XR games have opened a new arena for game industries. AR based game like Pokemon gained lots of popularity. There is a high potential in XR game arena

through web distribution. AR based game is the combination of 3D virtual objects and real environment. It needs to be ensured of proper mapping of 3D virtual object inside the real environment which is captured through device camera and these 3D objects should react as same as real objects beside them through rendering properly according to the camera position. And for the gaming purpose, these 3D objects should be interactive by human touch on the objects through the device display or console.

- AR/ VR application can be highly used in the education system to explain anything visually to the audience. 3D models (e.g., SketchFab), medical imaging, mapping, architectural pre-visualizations, and basic data visualization can help users to understand easily through augmented reality. But each time, for each scenario, it would not be justified to install different app for each purpose. But if we can serve AR or VR requirements through web browser, users will be able to experience and visualize it just by browsing the link. Therefore, AR/VR based education system is more appropriate through web distribution model rather than app store distribution model.

1.3 Objective

In this thesis, our main objective is to develop webXR applications focused on each problem those are discussed in the previous section. We will experiment with different type of XR applications which can be served through web technology instead of native app distribution model. Also, we will evaluate the performance of the webXR applications. To develop XR application for web, we will study on some open source APIs and frameworks such as webXR-device-API, webXR-polyfill framework, object detection libraries, 3D libraries etc. and will implement them in our webXR applications. There are different types of algorithms and libraries for real-time object detection. In this thesis we will also learn about them and will implement those object detection libraries in our webXR applications to compare their performance. But as our XR applications will run on the browsers, therefore, depending on the browser's capacity and support, our webXR apps can respond differently on different browsers, even some browsers may not support our apps at all. We will not focus on the browser's limitation in this paper.

1.4 Scope

We have developed 6 different webXR apps focusing on the issues described in the problem statement section to evaluate the possibilities, efficiency, and performance. In our first application, we have developed an XR app which is focused on "Detect and Control". This XR app is able to detect the television/monitor and users can control television or computer through the application. In the second app, we have checked the feasibility of implementing AR technology in the regular e-commerce website. In this

application, we have developed a basic e-commerce store of sunglass, where user can give a trial using AR functionalities. For both of these apps, OpenCV's haarcascade classifier algorithm is used for object detection. Another effective method for object detection is convolutional neural networks (CNN). In our third app, we have implemented TensorFlow's COCOSSD API (developed based on CNN) with webXR-device-API for real-time object detection. In this app, users can detect multiple objects in a frame and can tap on the name of the object to know more details about it. Another purpose of developing this app is to compare the performance with openCV. We have also considered the gaming arena in this thesis. In our fourth app, we have developed an AR board game with artificial intelligence, where player can play against computer. The fifth app is developed to show a demonstration with 3D model visualization for education sector. In our last app, we have developed another webXR game, where gaming element can be moved through object detection. 3D games require powerful processing unit to run efficiently, in other hand, real-time object detection algorithms also require high processing power. In this app, we have developed 3D webXR game with real-time object detection technique to evaluate the performance of the game. In the use-cases section [3], we have explained the concepts of each app in details.

1.5 Outline

In this section, we will give an overview of each chapter of this paper.
This thesis is organized with 6 chapters.

Chapter 2 explains about related works in this arena, the key technologies that are required to develop a webXR app. It describes different types of augmented and virtual reality, mobile based AR, web-based X Reality, Degree of freedom, and object detection and recognition techniques. In this chapter, we have also discussed about different frameworks and APIs that are used in our implementation.

Chapter 3 describes the different use-cases and the requirements to implements our XR apps. We have also described the concept and features of each XR application in details.

In **chapter 4**, we have explained logical concept of the application using the architecture of each app. We have also explained the concept using UML sequence diagram and flow chart diagram for some applications to provide a better overview.

Chapter 5 describes the functional implementation of the every application. We have explained some important part of the code here.

Chapter 6, describes the evaluation of our different application and features. In this chapter, we have explained our testing methods and testing results from different aspect of the applications. We have also shown the performance comparison of different

Object detection techniques in WebXR app.

At the end, **chapter 7**, describes the summery of the thesis, including the challenges that we have encountered during development.

2 Fundamentals and Related works

Till date there is one aspect of progressive WebXR, a version of A-Painter, that is already built to handle both AR and VR. To describe more, the progressive WebXR apps are usable for vast range of XR devices [PAR18]. There are such Android and iOS devices those supply augmented reality environment using which we are able to look at our surroundings and place various digital objects virtually in those visuals[CA18]
XR displays are either AR or VR based. Most common XR displays are Handheld or "Magic window" AR. It is supported by Apple's ARKit for iOS devices and Google's ARCore for the Android-based devices. These displays create an illusion for the clients turning their phones into a magic window that lets the client experience the AR. Headmounts such as Microsoft's Hololens provide users with a more convincing virtual surround experience. Similar devices are currently being developed or have already been released from companies such as Magic Leap and Meta.[PAR18]. As of now, XR displays uncover the stance (position and introduction) of the client's displays in the space around it. A few devices also help in tracking controllers in the same space while others sense the structure of that space, either as full 3D networks (on Windows MR and Hololens devices) or as planes (on ARKit and ARCore)[MAC18].
Article is a 3D model viewer which can be used for any kind of browser. On desktop, client will be able to see a 3D model. On the other hand, in mobile devices the experience is quite same: user has to touch and drag so that the model can rotate, or drag with fingers to zoom in. To elaborate, users can search the web, fetch a model and then place it in their room to look just how big really it is, and physically move around it[CA18].

In this section, we will discuss about the technologies those are required to develop an XR app for web. In this thesis, we are frequently discussing about Virtual Reality, Augmented reality. At first, we have explained AR and VR shortly. At 2.5, we have discussed about 'degree of freedom'. Then we have discussed about the key technologies and at the end, we gave an overview of frameworks and APIs, which have been used in our webXR application development.

2.1 Virtual Reality

The meaning of Virtual Reality is lying in its name. Virtual means near and Reality means what we experience as human being through our sense. So, the virtual reality is near reality, where the objects are virtually created but it feels like real. Virtual objects are generated by the computer. But what is reality, how do we feel something as real?

Before developing a virtual reality application, we need to understand the answer of this question. We all know that we have five most obvious sense organs, they are: sight (vision), hearing (audition), taste (gustation), smell (olfaction), and touch (somatosensation) [Wik18a]. But we have many other senses than these such as kinesthetic sense (proprioception), balance (equilibrioception) etc [Wik18a]. All of our sensory systems convey the information to our brain, and thus we consider reality. In other words, our entire experience of reality is simply a combination of sensory information and our brains sense-making mechanisms from that information [Vir17]. Hence, we can say that if we can manipulate our sensory system, or we can present our virtually created object and environment in such a way that our sense will respond accordingly, the brain will consider them as reality. So, in summary, virtual reality entails presenting our senses with a computer-generated virtual environment that we can explore in some fashion. [Vir17]. In the other word, when we experience the things through the computer, where the objects, the environments around us are virtual (computer generated), i.e. do not exists in reality and through our sense, it feels like real that is called virtual reality. The picture 2.1 (left) is a conceptual example of Virtual reality

According to the explanation we draw in above, it may feel that 3D movies or 3D PC games are also member of virtual reality because many times we lost ourselves in it, we feel many computers generated character as real. But they are not included in virtual reality.
In [Woo07], Chris Woodford explained that Virtual Reality must have some characteristics as follows

Believable: We should feel like, we exist inside the virtual world.

Interactive: We should be able to move around, able to look at any direction. 3d movies are not considered as a virtual reality because, in 3D movies, we only can see what it is viewing, we can not look at another side of the scene if we want.

Computer-generated: The environment and other objects can be created through computer 3D libraries or it can be a 3D recording of real world. But as this world will be presented in front of your eyes through computer, so the computer should be powerful with realistic 3D computer graphics, which are fast enough to present everything in real-time according to the user's movement to make it believable and interactive.

Explorable: The virtual world should be big and detailed, so that users can move around and explore it by themselves.

Immersive: To make it believable and realistic, we need to use human sense, therefore VR needs to engage user's body and mind. In the other word, while the virtual world is presented to the users according to their body movement and position, in the same time, mind also should sense it as real through its sound, sight etc.

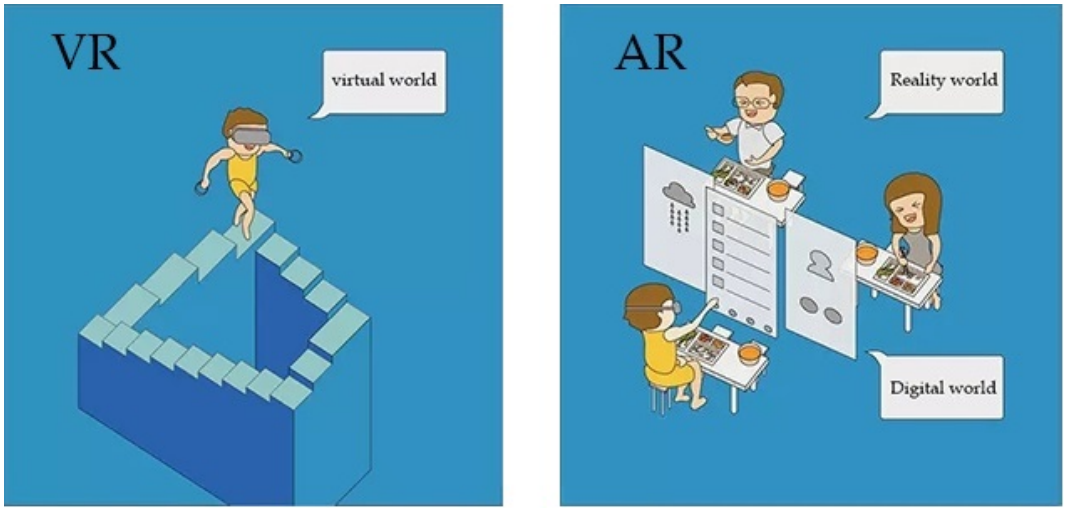

Figure 2.1: Conceptual example of Virtual reality vs. Augmented Reality [image source: imagenesmy.com]

2.2 Augmented Reality

The meaning of 'augment' is 'make something greater by adding something to it'. Augmented Reality and virtual reality are like two sides of the same coin. In Virtual Reality, the environment around us is created through computer-generated 3D graphics, where Augmented Reality uses the existing real environment using the device camera instead of computer generated environment. In Augmented reality, many computer-generated virtual 3D objects can be placed into the real world. Augmented reality is the combination of real world and virtual objects. The picture 2.1 (right) shows a conceptual example of Augmented reality, where users can view the real restaurant and in the same time they can see the food menu in front of them, which does not exist in reality

Based on use-cases and technologies, Augmented reality can be categorized in four sections[Rea18].
Marker Based Augmented Reality: It is also known as Image Recognition. In this technology, app uses camera and some kinds of visual marker such as QR code. When app detects any marker, it produces result according to the information lies with code. As the marker like QR code is simple pattern, therefore AR does not require high processing power. In the figure 2.2, we can see an example of Marker Based AR app.

Markerless Augmented Reality: It is also called location-based, position-based or GPS based Augmented Reality. In this technology, Augmented Reality uses digital compass, velocity meter, accelerometer, and GPS signal to provide data to the users.

Figure 2.2: Marker Based Augmented Reality [Rea18]

This type of AR applications is widely used for mapping direction, finding address nearby or to know information about nearby business. The picture 2.3 shows, how markerless augmented reality helps us to know information about the business around us. AR-enabled smartphones are commonly used to run this kind of application.

Figure 2.3: Markerless Augmented Reality [Rea18]

Projection Based Augmented Reality: In project-based augmented reality, AR app projects artificial light on the real-world surface. Users can interact or touch on the position of the light to do some action. As an example that has shown in 2.4, AR app projects some digits on the surface. When light projected on a surface, it receives a position, and when human touch on this light of digit, the light position changed because of human interaction. Detecting the difference between these two positions AR detect which digit has touched by the user and trigger the next action.

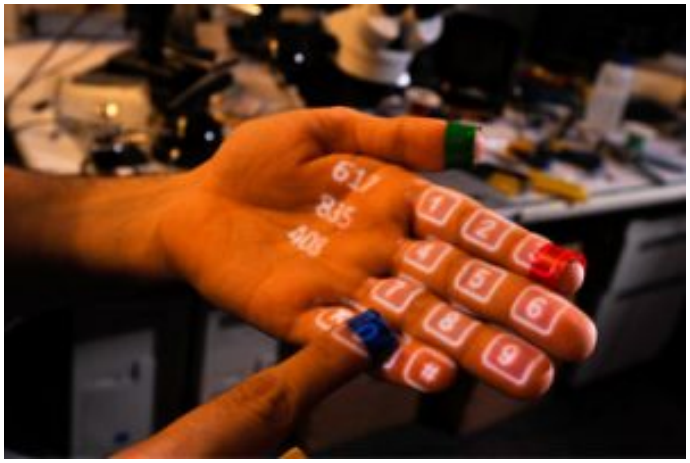

Figure 2.4: Projection Based Augmented Reality [Rea18]

Superimposition Based Augmented Reality: In this technology, AR places 3D virtual object into the real world as shown in 2.5. To place a 3D object in real world, app needs to understand the surface, position and other objects exit in the real world. As an example, users can place virtual 3D furniture in the room to visualize how it will look like in the room before they really purchase it. The app named 'Ikea place' is a great example of this kind of app.

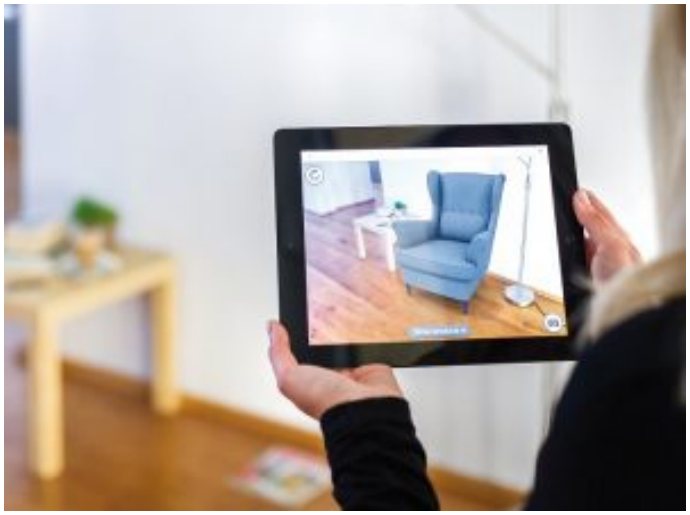

Figure 2.5: Superimposition Based Augmented Reality [Rea18]

2.3 Mobile augmented reality (MAR)

Implementing augmented reality technology to mobile devices such as iPhone, iPod, iPad or Samsung S9 etc., where real world is captured by cameras on mobile devices and mobile display is used for presenting the scene[YFL16]. There are many devices provided by different companies, which are expensive and only for AR or VR purpose. But nowadays everyone carries a mobile phone, therefore implementing AR application for mobile can provide the most benefit to the consumers. Because of rapid development in the mobile sector, mobile can meet all hardware and software requirements that need to run augmented reality app in mobile.

2.4 Web-Based X Reality

Web-based X reality defines when XR application i.e Augmented Reality or Virtual Reality application runs on the top of the web browser. XR applications are device specific, developed by targeting operating system like iOS, android or windows. Users need to install it before use. But accessing XR application through the browser removes all these limitations and it brings a lot of potential in different sectors such as e-commerce site, education sector etc. Developers need to develop only one app, which can be browsed through the web browser in any operating system on any AR-enabled mobile devices. But to develop XR application, app needs to track user's head position, orientation. Therefore. web browser needs to provide all hardware related information to the application. Web VR is supported by most web browsers like google Chrome, Mozilla firefox etc. But for Web AR, there are only few experimental browser exists: WebRonARCore (android), WEBRonARKit (iOS), Mozilla WebXR Viewer (iOS).

2.5 3DOF vs 6DOF

The elaboration of DOF is 'Degree of Freedom' and the number expresses the capability of hardware to detect the direction of movement in 3D space. 3DOF headset or mobile can track the head orientation only, i.e., in which direction users are looking at. According to [Sum11] the three axis in 3DOF are

- Tilting side to side on the X-axis. (Roll)
- Tilting forward and backward on the Y-axis. (Pitch)
- Turning left and right on the Z-axis. (Yaw)

But 6DOF headset or mobile tracks orientation with its position in the 3D space. 6DOF device knows, in which direction the user is looking at and where is the device in the space. The other three axis are [Sum11]

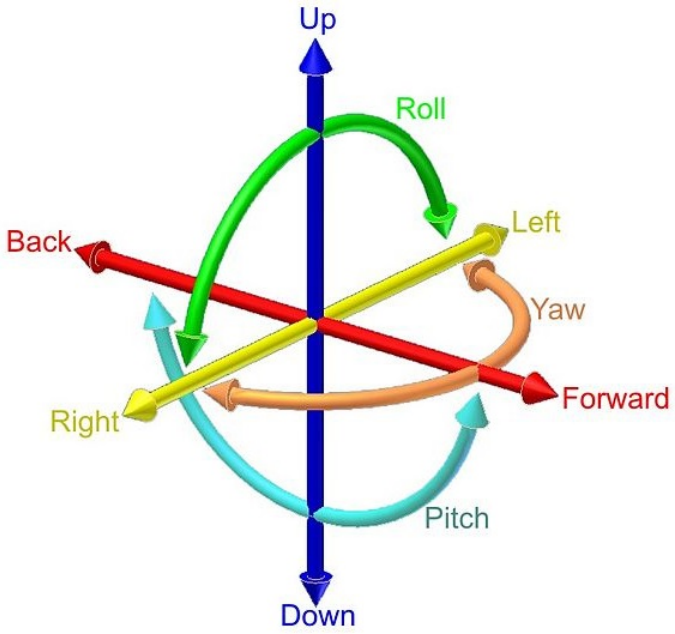

Figure 2.6: The six degrees of freedom: forward/back, up/down, left/right, yaw, pitch, roll[Wik18b]

- Moving forward and backward on the X-axis. (Surge)
- Moving left and right on the Y-axis. (Sway)
- Moving up and down on the Z-axis. (Heave)

The pic 2.7 shows how 3DOF device and 6DOF device works differently. The problem in 3DOF is, user can look around at any direction, but the position of the user in 3D space is fixed, therefore if the user moves the entire world moves with the user, users are not able to go near to any object. Every object is always at the same distance from the user. But in 6DOF, the environment does not move with the user. When user moves, the respective position of the user changes in the 3D world, hence user can go near to any object or can go far from it. In the picture 2.7 (left), we can see that, in both cases, user is in the center point of the 3D world. But from the figure from right, we can see that, when user moves to the left, in 3DOF, the whole world moves with it, but in 6DOF user moves near to the object.

360 video or some games such as car racing, where users do not need to move, always seat on a fixed position, 3DOF device works well with this kind of application. But for some other applications where users need to move around the environment, 6DOF enabled device is required. For Augmented reality app 6DOF tracking plays a vital role, as 3D object is placed in a fix position of the real world, therefore, when the device will move to any direction of the world, the virtual 3D object should be presented in the same way as any other real object beside it, is seen.

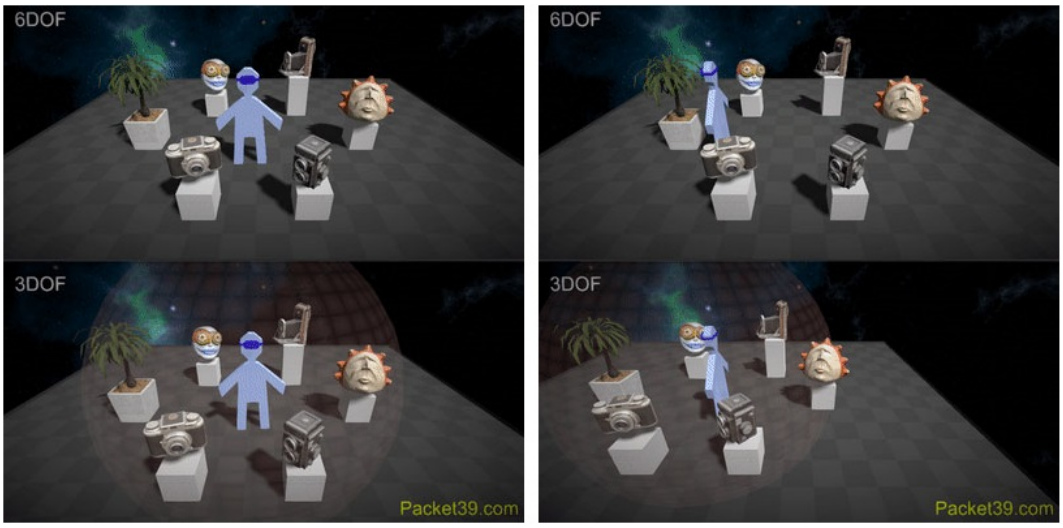

Figure 2.7: User movement 6DOF vs. 3DOF [Pac18]

2.6 Key Technologies

In this section, we will discuss about the required technologies that we need to run XR application on the devices especially in mobile. The key technologies are tracking and registration technology, object detection and recognition, calibration, model rendering, and display and interaction techniques.[YFL16]

2.6.1 Tracking and Registration Technology

To run AR application, it requires accurate orientation and position tracking to place the virtual objects into the real environment. When user will move the position or orientation, this technology should return the accurate position of the camera in real time, so that virtual object can be placed in an accurate position. When the camera position or orientation changes, the virtual object should be superimposed on the corresponding position, so that it looks like a part of the real scene.

As the name suggests, this technology has two steps: Registration, and Tracking.

Registration Process: Registration process happens when the virtual object is created and placed for the first time in real scene.

Tracking process: In this step, the virtual object is represented according to the camera position of the user. When user changes the position and orientation, system reconstruct the correct position between the virtual object and the real scene [YFL16]

2.6.2 Calibration

Calibration technology detects the orientation and position of the device. To provide the results accurately, system requires large number of data including sensor offset, camera parameters, the scope of the vision, object localization and deformation. Calibration technology measures these value and reports data to the system.[YFL16]

2.6.3 Model Rendering

Rendering is a process which generate 2D images utilizing 3D data and store the resulting image in a frame buffer. OpenGL ES library is used to process the rendering. OpenGL ES (Open Graphics Library Embedded System) is a 2D/3D lightweight graphics library, which is designed for embedded and mobile devices.[YFL16] To run augmented reality in browser, WebGL is used instead of OpenGL ES. WebGL is developed based on OpenGL ES 2.0 to provide maximize portability to mobile devices and to support rendering through web browser.[khr14]

2.6.4 Display and Interaction

This technology is the responsible for the final output which is presented to the user through the display of the devices. It controls how to display and interact with devices effectively and efficiently. Though mobile devices have small screen, poor computing power, but system needs to display smoothly and should response against every interaction effortlessly to make experience like real. As in mobile there is no keyboard or mouse, the interactive mode is just touch and swipe gestures.[YFL16]

2.6.5 Object Detection and Recognition

Object detection is a computer technology, which detects different objects such as human, buildings, cars etc. in digital images and videos. Object detection technology is the part of computer vision and image processing. Beside detection of the objects, this technique is also responsible for object localization. Object localization means detecting the position of the object in the scene. Each detection typically provides some form of pose information of the object: The position of the object is presented through the bounding box with x, y, width and height. In the picture 2.8, it shows different object with bounding box in different color.

In Augmented Reality, object detection and recognition are very important to provide highest possible benefit to the user. Note that, object detection and object recognition is not the same thing. Object detection technology identify the different objects from an image or video frame, but it does not describe the information about the object. Object recognition describes the name or type of the object.

In augmented reality the application should detect the objects automatically and it should display the information about it to the user. In the picture 2.8, there are different objects such as dog, bicycle, truck, tree etc. We can see that, red, yellow or green box

has been draw around the object which was detected by the system. On other side, it also defines the object name in green box as dog, the object in yellow box as bicycle and the object in red box as truck. This is called Object recognition.

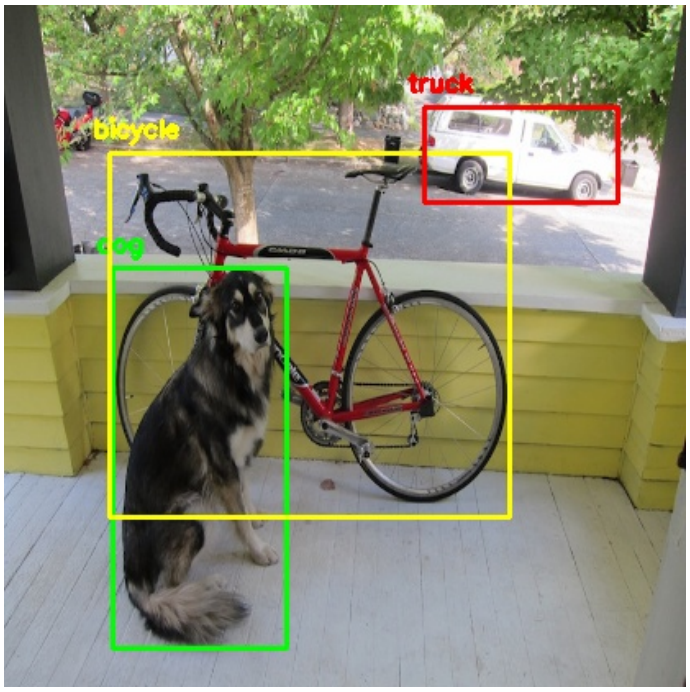

Figure 2.8: Object Detection and Recognition [Pon18]

There are many algorithms and technique to detect and recognise the object. In this thesis, we have used two different technologies in our application 1) Haarcascade object detection and 2) Object detection using Convolutional Neural networks. We have described them in brief at 2.6.6 and 2.6.7 accordingly.

2.6.6 Haar Cascade Object detection

Haarcascade is a machine learning object detection algorithm which is capable to identify objects in an image or video with high detection rate. The concept of this feature was proposed by Paul Viola and Michael Jones in the paper Rapid Object Detection using a Boosted Cascade of Simple Features in 2001 [VJ01]. Haar cascade algorithm has been implemented through four stages.

1. Haar feature
2. Create Integral Images to computer faster
3. Adaboost training
4. Cascading classifiers

- **Haar Feature:** This algorithm classifies image based on the value of simple features instead of using pixel directly, because feature-based system is much faster than pixel-based system [VJ01] There are three main features those are used in haar feature.

  Edge Features: This is also called two rectangles feature with white against black either in horizontal or vertical direction as shown in figure2.9. It computes a single value by subtracting the sum of pixel of white rectangle from the sum of the pixel of black rectangle.

Figure 2.9: Two Rectangles feature (Edge Feature) [Ber]

Line features: This is also known as three rectangles feature which is consists of dark in both side white or white in both side dark. This features also can be in both horizontal or vertical direction as shown in figure 2.10. It computes the difference between the sum of two outside rectangles and the sum of center rectangle.

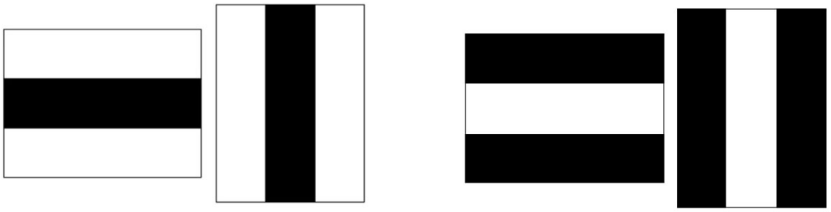

Figure 2.10: Three Rectangles feature (Line Feature) [Ber]

Four rectangles: It consists of two white and two dark as shown in figure 2.11. It computes the difference between diagonal pairs of rectangles.

All these three types of features generate a single value through subtracting the sum of pixels of white rectangles from sum of pixels of black rectangles. Let us consider an example shown in figure 2.12 (a), where 0 represents white pixel and 1 represents black pixel.

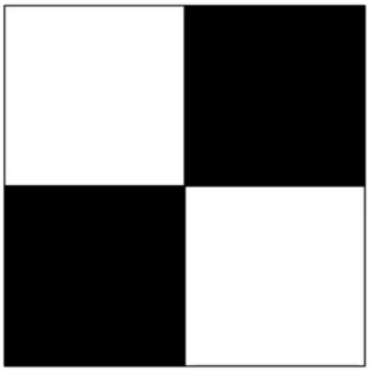

Figure 2.11: Four Rectangles feature [Ber]

So, its edge feature = mean(sum Of black) - mean(sum Of white) = 1 - 0 = 1

But in real scenario, our images do not have perfect edges or lines. If we transform any color image to black and white (gray scale), for a part of the image we will receive pixel like 2.12 (b)

So, mean of Sum of Black =

$$\frac{(1 + 0.9 + 0.8 + 0.8 + 0.9 + 0.8 + 0.7 + 0.8)}{8} = 0.8375$$

And mean of Sum of White =

$$\frac{(0 + 0.1 + 0.1 + 0.1 + 0.2 + 0.2 + 0.2 + 0.1)}{8} = 0.125$$

it's edge feature = 0.8375 - 0.125 = 0.7125
The result as much closer to 1, it is considered as better feature. To detect haar feature, we have to search for the feature all over the image. Hence it needs to calculate the sum of the pixel value for each possible area, which is very expensive to compute. To solve this problem Viola-Jones[VJ01] proposed an algorithm which is called integral image. It allows to evaluate feature very fast.

- **Integral Image:** Integral image is the intermediate representation of the image, which allows to compute rectangle features very fast. In the integral image, the value at location x,y is the sum of the pixels above and the left of the location x,y. It can be represent as

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad \text{[VJ01]}$$

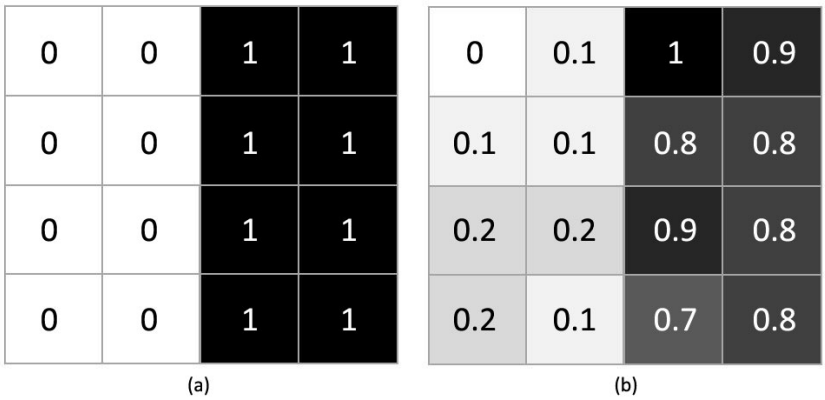

Figure 2.12: Two Rectangles feature Calculation Example

In the figure 2.13, the value of integral image at location 1 is the sum of the pixel values in rectangle A, the value at location 2 is the sum of A and B. The value of location 3 is A+C and at location 4, the value will be sum of A + B + C + D. If we want to calculate only the sum of the pixel value within D, we can compute as 4+1-(2+3) [VJ01]

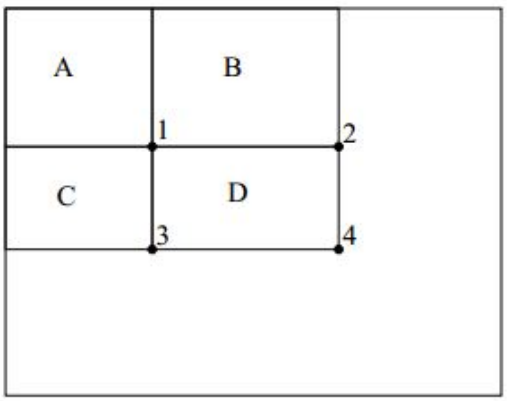

Figure 2.13: Integral Image [VJ01]

Let's calculate with a real example. Figure 2.14(a) shows the each pixel value of an image. So the intermediate integral image can be represented as 2.14(b). Now consider pixel value of any x,y location. Say, the pixel value at location [0,0] is 31, and as in main image the pixel value at location [0,1] is 2, therefore, in integral image, the pixel value at location [0,1] will be 31+2=33, at location [0,3] pixel value will be 31+2+4= 37. In the same way in the next row at the location [1,0], the pixel value will be 31+12=43, the value at location [1,2] will be 31+2+12+26+13+17=101 in integral image.

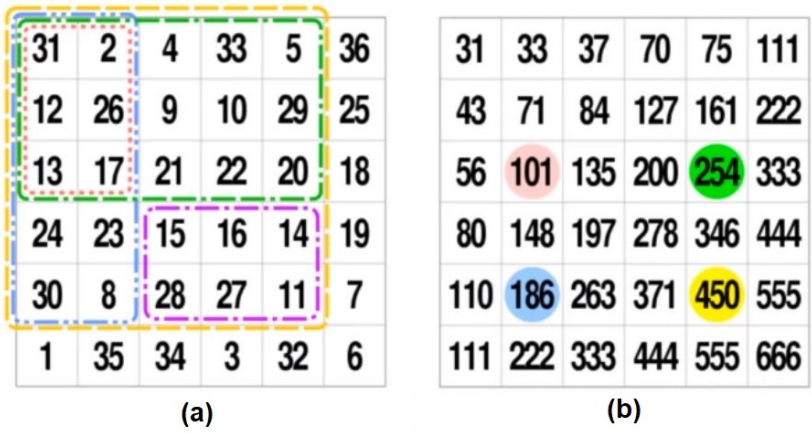

Figure 2.14: Integral Image Example [Ber]

Now, if we need to compute the sum of the pixel value for the rectangle between [0,0] to [1,2], we can get the value 101 directly. In same way if we want to calculate for the rectangle between [2,3] and [4,4], according the equation mention for figure 2.13, we can calculate it as 450+101-254-186 =111, which is same as sum of the pixel values of that rectangle of the origin image i.e. 15+16+14+28+27+11 = 111. In this way, we can compute the sum of a rectangle very fast using integral image which allow us to detect haar feature.

But there might be over 18000 rectangle features associated in each sub window of an image. This number is very larger than the number of pixel[VJ01]. According to [Ope18b] more than 160,000 features can be detected in a 24x24 window of an image, where all of them may not be relevant to detect any object. But combining a very small number of features, an effective classifier can be created. To achieve this, adaboost algorithm are used.

- **Adaboost training:** Adaboost is the short form of 'Adapting Boosting', which was proposed by Freund and Schapire in 1996. The aim of adaboost is to convert a set of weak classifier into a strong classifier. The equation for classification can be represented as

$$F(x) = sign(\sum_{m=1}^{M} \theta_m f_m(x)) \quad \text{[Tow18]}$$

where f_m is the value for m^{th} weak classifier and $theta_m$ is the corresponding weight. The result is the weighted combination of M weak classifiers. [Tow18]

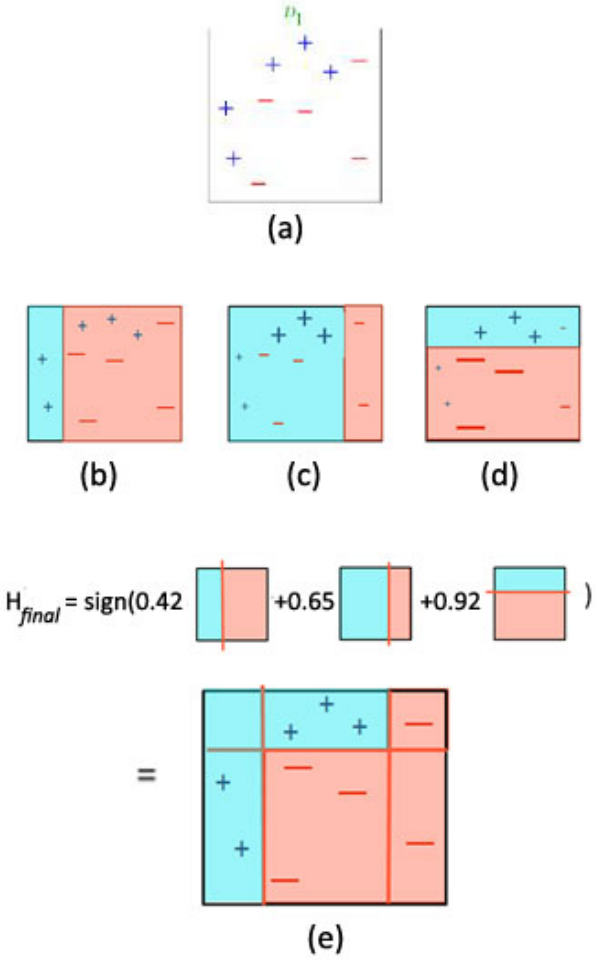

Figure 2.15: Adaboost Example [Sin12]

Lets consider a data set 2.15(a), where we have to classify + and - data. At first run the weak learning algorithm without any weight which results initial classification as shown in figure 2.15(b). We can see there are three + data, which are classified in wrong region. This is one of the weak classifier. Now we will run the weak learning algorithm again, increasing the weight of those miss classified three + data. The result looks like figure 2.15(c). In this result, algorithm made sure that the weighted data is in the right region. But to achieve this, we can see there are other three '-' data, which are in the wrong region. So we will run the weak learning algorithm again by increasing weight of those three '-' data, which give result shown as 2.15(d).
Now we have three weak classifier, who are partially wrong. Using this set of weak classifiers we can convert them into a strong classifier 2.15(e)

In Haar cascade training, a feature set and a training set of positive and negative images are provided for classification. In this system, a variant Adaboost is used to select both a small set of features and training classifier[VJ01]. There are a big number of features can be detected, where all of the features are not useful to build up a strong classifier. So, another challenge is to find a small number of features, which are best to build a strong classifier. To achieve this, the weak learning algorithm is used which is capable to select rectangle feature that separates the positive and negative examples in best way. For each feature, the optimal threshold is determined by the weak learner, so that minimum number of examples are miss-classified. As each haar feature is just a "weak classifier", so a large number of haar features are organized into cascade classifier to build a strong classifier so that an object can be detected efficiently.

The figure 2.16 shows an example of haar features for a face. Eyebrows in human face are darker than the lower region, so it can be an edge feature (2 rectangles haar feature) for a face. In the same way top of the nose is lighter than the both side of the nose. It can be line features (three rectangles feature). But the features for eye and nose region can be detected in other way. In the figure 2.16, the first haar feature, eye region is the darker than the nose and cheeks region. And in second haar feature both eyes are darker than the bridge of the nose. Same kind of edge or line features may present cheeks or any other place, which is not relevant. So, these two features can be considered as best features among all other haar features.

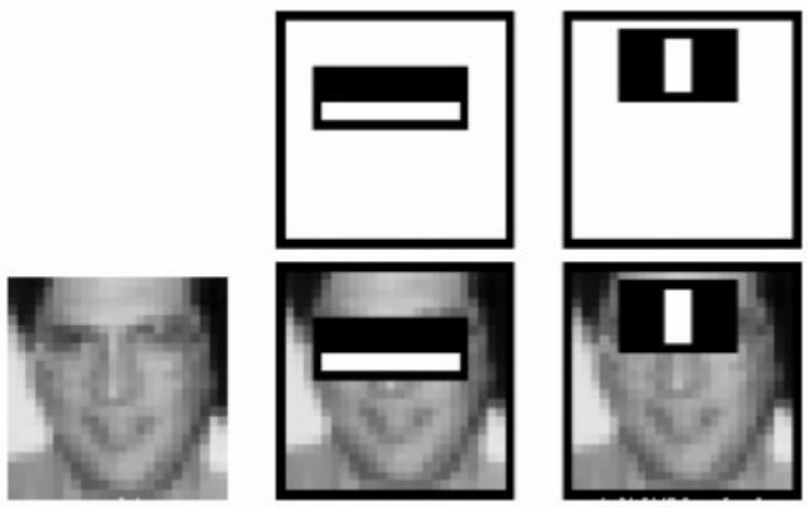

Figure 2.16: Haar Cascade Feature [Ope18b]

- **Cascading classifiers:** The algorithm of a cascade of classifier is responsible for increasing detection performance with reducing computation time. It is a collection of stages where each stage is a result of weak learners. Boosting technique is

used to train each stage to achieve highly accurate classifier. It takes a weighted average of the decisions made by weak learners. During the detection process in each stage, all sub-windows are labeled as either positive or negative. If an object is found, the sub window is labeled as positive and if no object is found, the sub window is labeled as negative. If a sub window is labeled as positive, the classifier passes that region to the next stage, so that in next stage classifier can be adjusted to achieve high accuracy rate. But if the sub window is labeled as negative, the region is immediately rejected and the classification for the region is considered as complete.

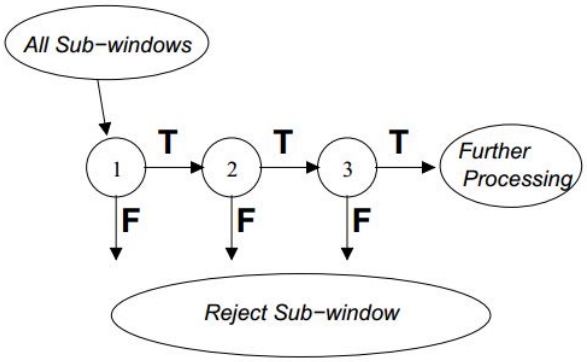

Figure 2.17: In every stage sub-windows with negative label are rejected [VJ01]

In the figure 2.17 shows that how in every sub-windows with negative labeled are rejected and only the positive sub-windows reach in next stage. In an image majority of region does not contain the object of interest. Cascade attempts to reject highest possible negatives in early stages. After some stages, the number of sub windows are reduced. But during this labeling process, different types of mistake can take place. Such as a sub window which should be labeled as positive can be labeled as negative mistakenly. So the occurrences are classified in three ways.

- True Positive: It occurs if a positive sample is correctly classified.
- False Positive: It occurs if a negative sample is classified as positive.
- False Negative: It occurs if a positive sample is classified as negative.

Each stage should have low false negative rate. If in any stage positive samples are classified as negative, as it is immediately rejected, therefore this mistake can not be corrected anymore. But each stage may have high false positive rate, which can be corrected in next stages. Therefore, adding more stages can reduce the false positive rate.

For Haar cascade Training, a set of positive images and a set of negative images are provided. Positive images contain the object of interest and in negative images

do not contain the object of interest. Positive samples are created through positive images and negative samples are created using negative images. User must provide number of stages, feature type and other functional parameters. After each stage of training, final classifier is created. OpenCV has inbuilt feature of haarcascade training from which we can create our own classifier for an object very easily. We have discussed about the command for haar cascade training in 2.7.5

2.6.7 Object detection through Convolutional Neural Network (CNN)

In deep learning Convolutional neural network is another method to analyze visual image. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex [MMMK03]. CNN is a very powerful method in image processing and artificial intelligence, which use deep learning for object detection and recognition in image or video. Convolutional neural networks are like regular neural Networks with little change in the architecture. In regular neural network, an image transformation happens by putting it through a series of hidden layers. Hidden layer is a layer between input layer and output layer, which is a set of weighted inputs to produce an output through an activation function. Finally, there is fully connected layer and output layer to predict the objects in the image.
Suppose we have an image 2.18 with 1000 x 1000 resolution which have 3 channels of RGB values. Therefore, the dimension of our input images will be 1000 x 1000 x 3 = 3-million. To analysis this image, we may have hidden layer with 1000 units, which mean 3 billion parameter needs to be computed to forward it for the next layer.

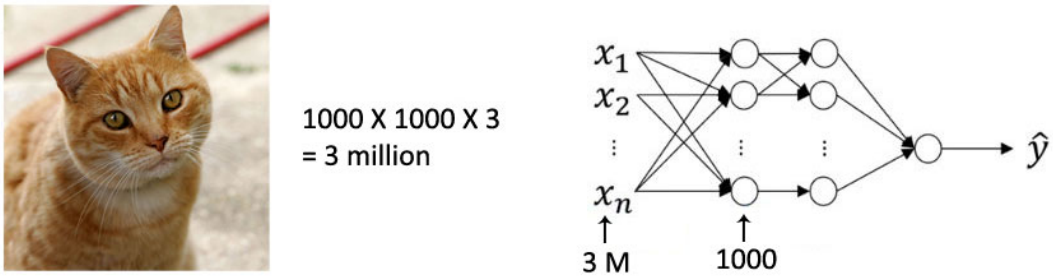

Figure 2.18: Example of regular neural network

The computation and memory requirements to train a neural network with these large number of parameters are really very high, which is not feasible. The image can be larger than the example. To solve this problem of regular neural network, convolutional neural network takes place.
In convolutional network, the layers have 3 dimensions, width, height and depth. Furthermore, the neurons in one layer compute only a small region of the layer instead of total layers. In the figure 2.19

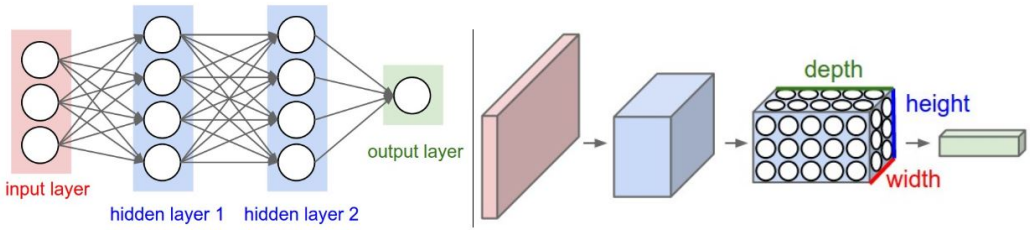

Figure 2.19: Regular Neural Networks Vs Convolutional Neural Networks [Kar12]

Layers of Convolutional neural Networks

A Convolutional neural Network is a sequence of layers, where output of each layer is the input of next layer. There are mainly three types of layers in a CNN architecture: Convolutional Layer, Pooling layer and fully-connected layer. Each layer type can be used multiple times to generate best output.

1. Convolutional Layer: This layer is responsible for convolution operation on an image. It is one of the fundamental building blocks of a convolutional neural network. In an object detection algorithm, most basic operation is to detect features of the image, then later layers may detect cause of objects from the feature and finally some later layers can detect complete object like person, cat, dog etc.

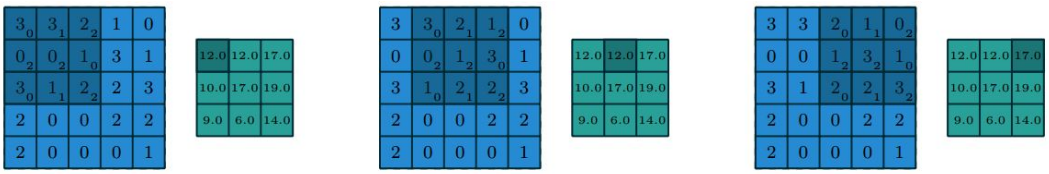

Figure 2.20: Convolution operation [Mur16]

Figure 2.20 shows, how convolutional operation run on an image, Where blue plot is the image, dark blue subplot indicate neurons in the filter region. Green image is the output of the convolutional operation, where dark green is the output of the filter region.

To understand how convolutional network works, we will go through an example where we will detect the vertical edge feature. As shown in the figure 2.21 (a), suppose we have a gray scale image, where left half of the image is white and right half of the image is black. Therefore, the vertical edge is located at the center of the image vertically. As the image is gray scale, so it will be 6 x 6 x 1 matrices. (for RGB it would be 6 x 6 x 3 because of it's 3 channels). To detect the edge, we will run a convolution operation with 3 x 3 x 1 filter matrix which looks like [1,1,1,0,0,0,-1,-1,-1]. As a result we will receive a 4 x 4 x 1 matrix. The way the convolution operation works is,

the 3 x 3 x 1 filter cooperate with the first region of 3 x 3 x 1 of the 6 x 6 x 1 input images (blue region) and get a value for the region. According to our image it will be like

$$10x1+10x0+10x(-1)+\ 10x1+10x0+10x(-1)+\ 10x1+10x0+10x(-1) = 0$$

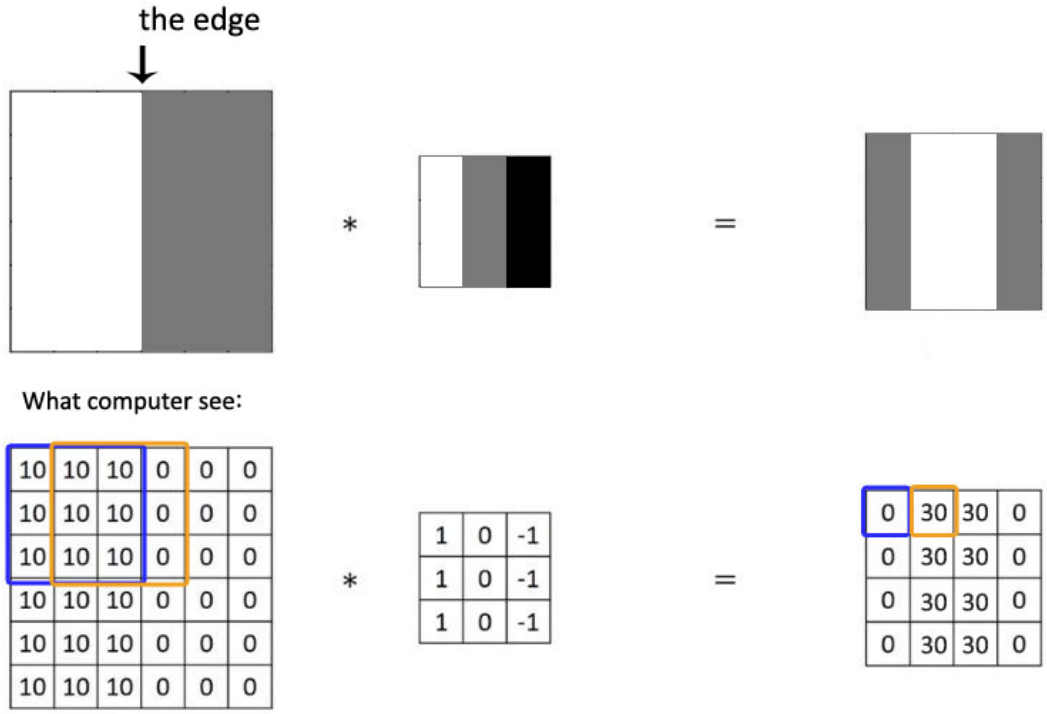

Figure 2.21: Vertical Edge Detection using CNN [Ng]

So, 0 is the value for the [0,0] index of the output layer. Then it shifts by one column and compute again. If after shifting by a column, the region of input layer does not match with filter matrices, it shifts by one row from very left column and continues in the same way. At the end, it generates the output layer with 4 x 4 matrix, from which we can detect the edge. In this example (fig 2.21), from output layer we can see the edge of the image is in center position. Though it looks big edge, but it works well when the image is larger.

The problem with such convolutional method is, the pixels at the border of the image are not computed as same as the pixels in middle of the images. Therefore, we lose the information of the border of the image. To keep more information of border, padding is used.

Padding: Padding basically added extra pixel with zero value which are placed around the image. Padding allows us to use a convolutional layer without shrinking the width and the height of the volumes. Figure 2.22 shows the example of padding. Depending on the padding value, we can divide convolutional operation into two categories: Valid and Same convolution.

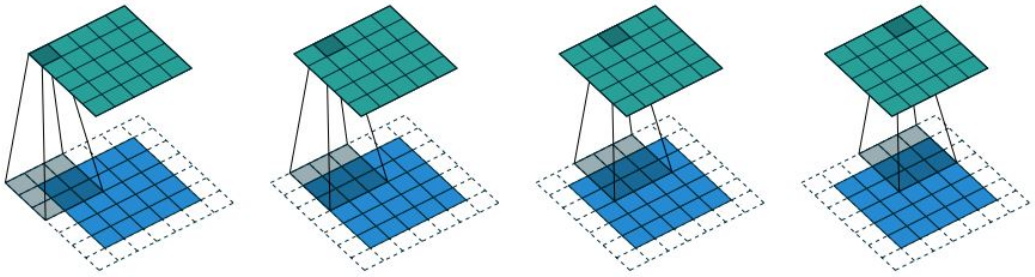

Figure 2.22: Example of padding [Mur16]

In *Valid convolution*, no padding is used, where, in *Same convolution*, padding value is chosen in such a way, so that the output size is the same as the input size.

The formula for convolution layer can be presented as
If the input image is: n x n
Filter image is: f x f
Output image size will be: (n+2p-f+1) x (n+2p-f+1)
So according to our example image 2.21, our output image size was (6+2.0-3+1) x (6+2.0-3+1) = 4 x 4
So, if we want to choose the padding for same convolution, we can formulize it as

$$if, n + 2p - f + 1 = n$$

$$padding, p = \frac{f-1}{2}$$

Stride convolutional: According to our previous example, during running convolution operation of each region, filter matrices were shifted by one row or column. In stride convolutional operation, instead of shifting one row or column, we can shift multiple rows or columns by setting the stride value. If we use the stride convolutional operation, the formula for output image size can be updated to

$$\frac{n+2p-f}{s} + 1 \times \frac{n+2p-f}{s} + 1 \qquad \text{[Mur16]}$$

But, the result should be an integer value after choosing the value for stride, otherwise when filter matrices will not fit at border region and it will skip border pixel computation. So, if the value of (n+2p-f)/s + 1 is floating value, we should decrease the value

of s to get an integer result.

Convolution operation over volume with multiple filter: In the previous example, the image was considered as gray scale image with n x n or n x n x 1 matrices. But an image may have multiple channel which can be represented as $\times n \times n_c$, where n_c = number of channels. In case of RGB image, the number of channels are 3, so input matrices should be n x n x 3. To run convolutional operation, the filter layer also should have the same number of channels i.e., $f \times f \times n_c$. In output, the value of all channels for each position are added, therefore the output for each filter have one channel.
To detect an object, we may need to apply different types of filters. The output matrices of each filter are added as a channel for output layers. So the channel of output layers depends on the number of filter. The formula can be represented as

$$(n \times n \times n_c) * (f \times f \times n_c) = (\frac{n+2p-f}{s} + 1) * (\frac{n+2p-f}{s} + 1) * n'_c \ ,$$

where,
n_c = number of channels
n'_c = number of filters

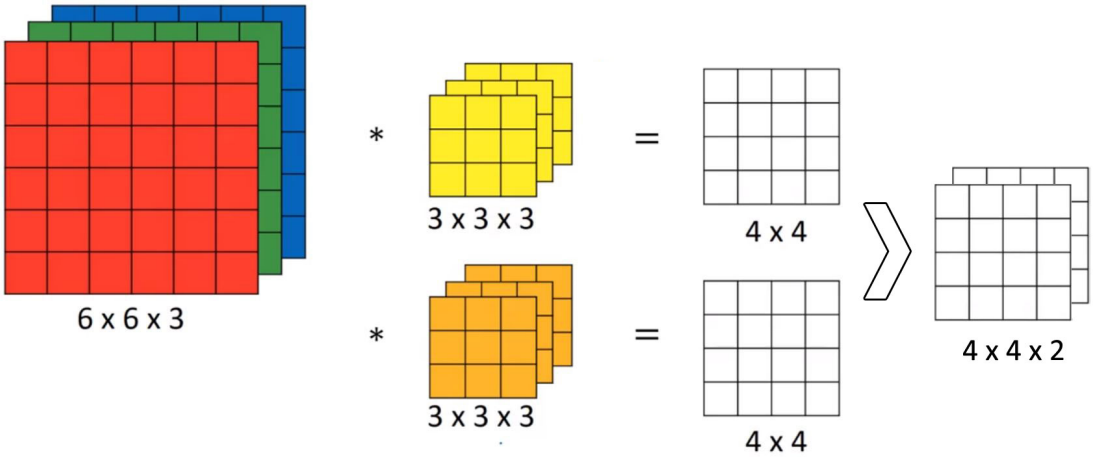

Figure 2.23: Multiple filter with multiple channels [Ng] (modified)

In the figure 2.23, we have 6 x 6 x 3 input layer, and suppose we would like to apply two types of filter (as example, for vertical and horizontal edge detection). The output layer will be 4 x 4 x 2

As, each layer will be convoluted with previous layer, so if layer l is a convolution layer, then

Input layer= $n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$, where h = height, w = width

Lets consider,

Filter size: $f^{[l]}$

Number of filter: $n_c^{[l]}$

Padding: $p^{[l]}$

Stride: $s^{[l]}$

So each filter: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Output: $n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$, where $n_h^{[l]} = \frac{n_h^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1, n_w^{[l]} = \frac{n_w^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$

Activation: $a^{[l]} = n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

[Ng]

2. Pooling Layer: Convolution networks use pooling layers to reduce the size of the representation, the amount of the parameters and to speed up computation. Pooling layer generally reduce the size using Max pooling operation. Most commonly max pooling operation run on each 2 x 2 (width and height) region of the input layer with a stride value of 2. So max operation done among 4 numbers. But depth remains unchanged in this layer. Suppose,

The size of the input layer is $w_1 \times h_1 \times d_1$

Region for max operation: f

Stride: s

So the size of output layer will be: $w_2 \times h_2 \times d_2$ [Kar12]

where,

$w_2 = \frac{w_1 - f}{s} + 1$

$h_2 = \frac{h_1 - f}{s} + 1$

$d_2 = d_1$

Let's consider, (figure 2.24 (b)) our input layer is $w_1 \times h_1 = 4x4$, where we will execute the max pool operation with f = 2 and s = 2. From the first pink region, the maximum value among the four value is 6, so it will be the first value of the output layer. In the same way, this operation finds the maximum value of each region. And as a result, we receive the output layer with 2 x 2. The figure 2.24 (a) shows, how the size of representation is reduced through pooling layer.

In practice most common used parameter for pooling layer is f =2, s = 2. But overlapping pooling layer is also used sometimes. Most common parameter for over lapping pooling is f = 3, s = 2.

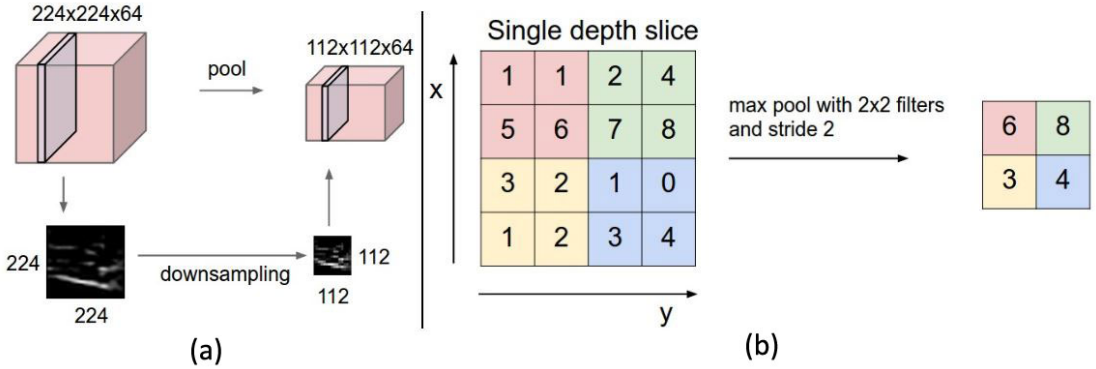

Figure 2.24: Max pooling in convolution neural networks [Kar12]

There are another pooling method, which is also used in pooling layer. It is called *Average pooling*. In this method instead of taking the maximum number, it takes the average value of the numbers.

3. Fully Connected Layer: After convolution layer and pooling layer, the next task is classification of the image. This classification task is done through few fully connected layers. Fully connected layers connect every neuron in one layer to every neuron in another layer. These fully connected layers only can accept one Dimensional data. But as the pooling layers are multidimensional matrix. Hence, the output from pooling layers need to be converted to the one-dimensional vector before connecting it with fully connected layer. This process is called "flatten". Suppose, we received an output from pooling layer with 5 x 5 x 16, So after flattening, it will be 400 x 1 vector. This flatten layer connects with fully connected layer where every neuron connects with every neuron of next fully connected layer. Each neuron holds a number, which is usually between 0 to 1. This number inside the neuron is called activation. Each fully connected layer makes some activation depending on the edges or patterns. Bias parameter is used as weight to control the activation of the neurons for the next layer. And finally, fully connected layer connects with softmax layers. Softmax layer is the list of class that to be predicted. If we predict between cat or dog, softmax layer will have two neurons. If we predict the handwriting number among 0 to 9, softmax layer will have to 10 neurons. At the end based on prediction value, we may detect the object.

The figure 2.25, shows an example of entire convolution networks, where we have an image with 32 x 32 x 3 that contains number 7, with RGB channels. In first convolutional layer, if we use the parameters as 5 x 5 x 3 filter, no padding, stride = 1 and 6 different filters, the output will be 28 x 28 x 6. Now in next pooling layer, if our parameters are as filter = 2 x 2 and stride = 2, the output will be 14 x 14 x 6. In the same way, we can run convolution and pooling layers several times for better result. In this example we run two times, where we received an output layer with 5 x 5 x 16 after the second pooling layer. After flatting we will receive a vector with 400 neurons. Now during

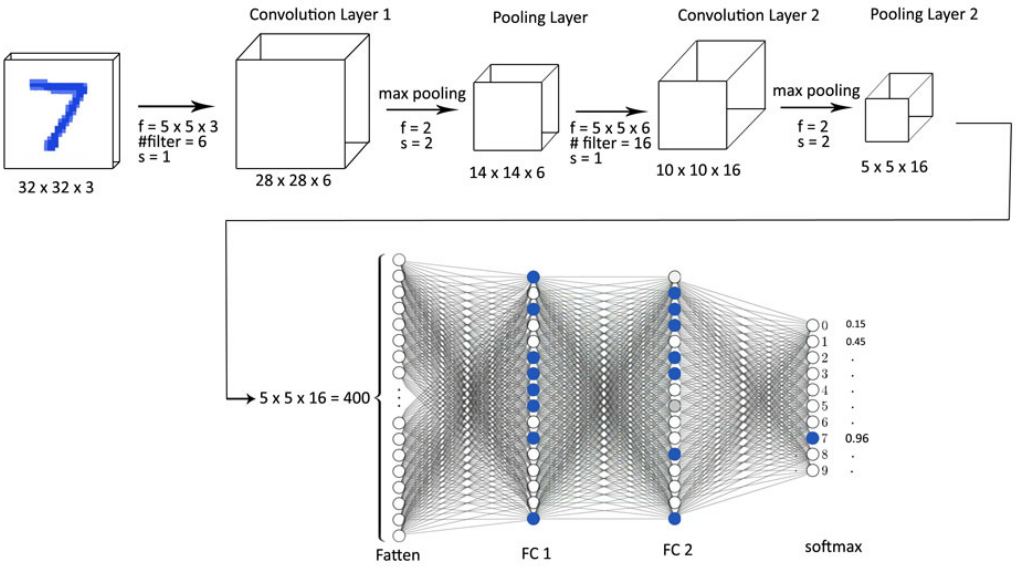

Figure 2.25: Example of Convolution Neural Networks [Ng] (modified)

going through some fully connected layers it will activate some neuron, which will give a prediction for every neuron at softmax layer. So at softmax layers, every neuron will have an activation number, which is the prediction value between 0 to 1, where 1 is the highest possibility and 0 is the lowest possibility. In the figure, we can see the system detect as 7 with 0.96 possibility, where the possibility for other number is less than 0.96. So, from the prediction value, we can declare the number in the image is 7.

2.7 Frameworks and APIs

To develop a VR or AR app, developers need to access accelerometers, gyroscope, gps location, digital compass etc. to get accurate position and orientation of the camera. But third party developers does not have access on the hardware directly. Operating system of a device provides different type of hardware related data such as touch, camera, rotation, position, orientation etc. to developers through different frameworks to build application for that operating system.

In this section, at first we will discuss about such kind of framework such as ARKit and ARCore which provide us required data to build native XR-application respectively for iOS and android. In thesis, our main goal is to develop an web-based XR application. Our application is not the native application for an operating system, rather the application will be hosted in a server and accessible through the XR-enabled browser from any operating system. When developers develop an application targeting an operating system, mostly native programming language is used, so that all hardware related infor-

mation can be directly accessible inside the code. As example, to develop an application for iOS, Objective-C or Swift is used as programming language, as it allows developers to build the app as native application and all the libraries, framework that provide information regarding hardware such as touch, accelerometer, network protocol etc. are accessible through Objective-C or swift, In same way, Java is the native language to build an application for android. Sometimes to keep same base code, developer use C++ which run top of the native languages and communicates with framework through it. But as our application should run through the browser, so HTML, CSS, javascript etc will be used as programming language. Hence we need a framework, through which we will be able to communicate with ARKit (2.7.1) or ARCore 2.7.2 to access the information of the device hardware. WebXR-Device-API is the framework, which establish communication between web-application and Device. In this chapter, we will discuss in details about WebXR-Device-API, the WebXR-Polyfil framework. Then we will discuss about other frameworks such as openCV, tensorflow, threeJS, smasung API, robotjs etc. which we used with WebXR api to build our XR-application.

2.7.1 ARKit

ARKit is a framework which is developed by Apple to enable developers to build augmented-reality apps for iOS devices i.e iPhone, iPad. ARKit performs mapping as device is moved using device's camera, accelerometers, gyroscope and provide data to developers to build their own AR apps. ARKit was first introduced with iOS 11.0. Hence, AR application requires iOS 11.0 or later and iOS device should have an A9 or later processor. Some ARKit features require later iOS versions or specific devices[App18]. As an example, in iPhone-X, ARKit can perform real-time face scanning and analyzing the face expression, the resulting data is used to change facial expression of 3D character.

2.7.2 ARCore

ARCore is a framework developed by Google to build augmented reality app for android devices. ARCore enables android phone to sense its environment, understand the world and interact with this information.[Goo18]

2.7.3 WebXR Device API

WebXR device API (also known as WebXR api) is developed by Immersive Web Community, which offers an easy communication between web-based XR app and the hardware. It gives access to input and output capabilities commonly associated with Virtual Reality (VR) and Augmented Reality (AR) hardware[Imm18a]. An XR app can be run in different kind of hardware such as Google's Daydream, Oculus Rift, any VR/AR enabled mobile. When a developer develops an app, he/she does not need to think about the appropriate hardware. Developers can simply access required hardware information,

status, state, position etc. through webxr api.

According to [Imm18a], webXR-device-API is in development to serve the following purposes for a XR application:

- It detects if the device has XR capabilities or not
- It interrogates the capabilities of XR devices.
- Examine the XR device and the state of associated input device.
- Display the application view on the XR device with all XR functionalities at appropriate frame rate.

But webXR-device-api does not talk about how the XR enabled browser should looks like or the features of the browser.
Though at the beginning it was assumed that, webXR api will be used for game development, but later it was found that, webxr-api has more potential to serve other purpose as well. There are many use cases for AR / VR application, which does not fit in app store model rather they are more appropriate for web distribution. WebXR device api is aimed to serve these purposes, so that through the website, device can provide XR experience to the users like a native app.
In this section we will discuss WebXR-device-API in details.

Lifetime of XR web app

According to the [Imm18a], the life time of a XR web app can be explained as shown in figure 2.26.

When XR enabled webpage loads into the browser

1. API checks, if the device supports desired XR mode
2. If the device supports required XR mode, application display the XR Mode option to the users
3. If user request for XR mode, i.e tap or click on the XR Mode option, API request an immersive XR session from the device.
4. API runs a render loop to produce graphical frame using this session and this graphical frame is displayed on the XR device screen.
5. This render loop continues to produce frame until users wish to exit XR mode.
6. If user tap or click on exit XR Mode, XR session ends

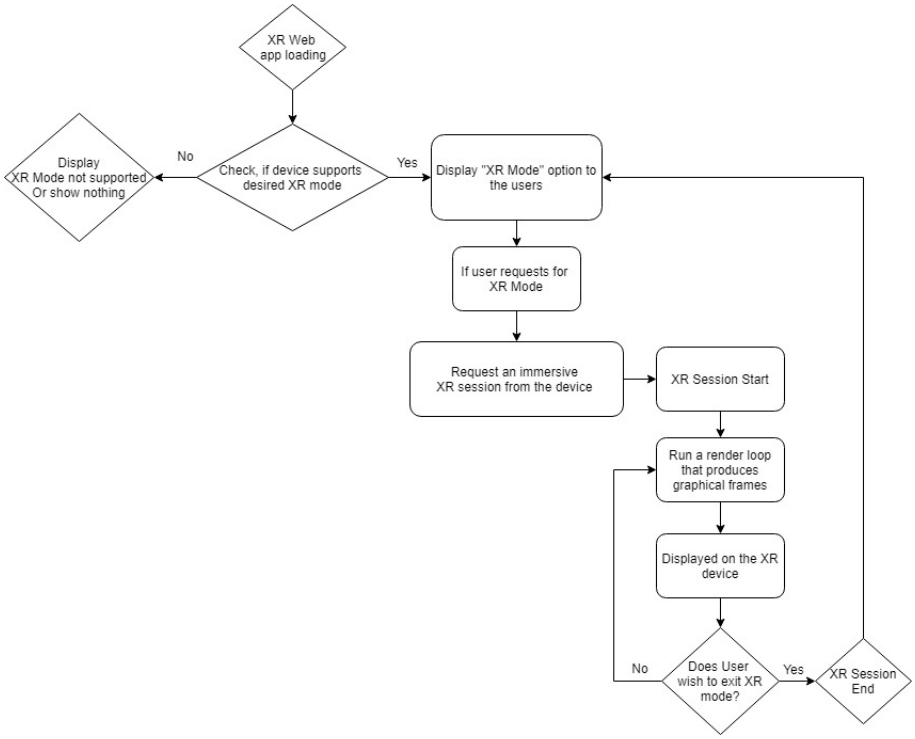

Figure 2.26: Lifetime of a XR web app

XR Object

XR object is the entry point to the webXR device API.[Imm18b]. When XR web app is loaded, XR object is initialized. During initialization of XR object, it checks if the deice supports desired XR features. If the required features are available, it creates a list of XR devices (explained at 2.7.3:XRHardware), which must be initially empty. Then it creates XR Session (explained at 2.7.3:XRSession) and initiates communication with XR hardware via XR session.

XR Hardware

XR hardware is a physical unit which is able to present imagery to the user. It is also referred as "XR device". If the XR webpage is run on desktop, a headset that connected with desktop can be considered as XR device. Mobile device itself is XR device, where for Augmented reality the mobile display can work as XR device, or using mobile with other viewer hardware such as google cardboard, daydream or Samsung gear VR, users can experience virtual reality or augmented reality. When XR web app will be run in desktop, it is also possible that at starting no XR device is connected with desktop, but later user connects it or at starting XR device was connected but later when XR app is running, XR device becomes unavailable. Hence API always listen to the devicechange event emitted on navigator.xr to respond immediately on change of the device availabil-

ity. In figure 2.27, shows the steps if devicechange event is trigger. This figure has been developed according to the description [Imm18a] to provide an easy overview.

```
navigator.xr.addEventListener('devicechange', checkForXRSupport);
```

[Imm18a]

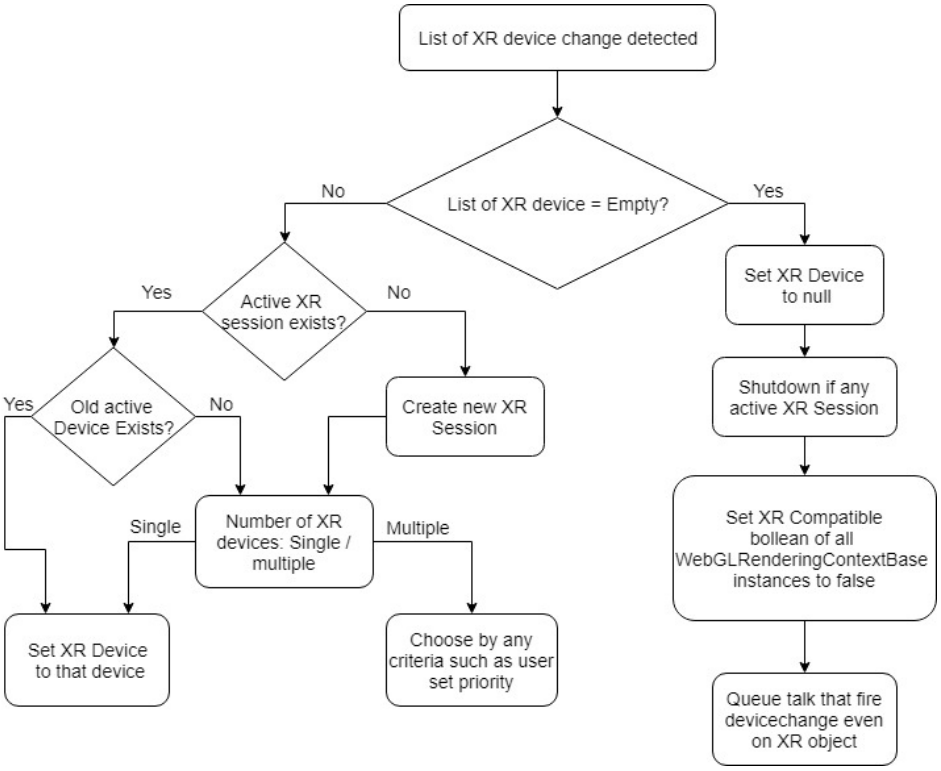

Figure 2.27: The steps when list of XR device changes

XR Session

XRSession establish communication with XR hardware. Through XR session, it is possible to get any hardware information such as device position, orientation etc. XR session can be retrieved by calling requestSession() on XR object. When user agent requests a XR session, XR object checks if the required XR features are supported by the device or not by using the function navigator.xr.supportsSessionMode. According to the availability of XR features, XR mode is initialized. If the device does not support any kind of XR features, which can be justified by any XR mode, XR object rejects the request of XR session creation.

There are three types of XR modes

- **Inline:** In Inline session mode, XR session does not able to display contents on the XR. It only provides access to receive device tracking information, which can be used for the rendering the content of the page. Inline is the default mode. It also can be requested by specifying the enum value as 'inline' [Imm18a]
- **Immersive VR:** It can be requested with enum 'immersive-vr'. In immersive VR session mood, XR session provides access to the XR device display.[Imm18b].
- **Immersive AR:** It can be requested with enum 'immersive-ar'. Like immersive-vr, in this mood, XR session provides access to the XR device display, but the content is intended to be integrated with the user's environment. For immersive-ar session, the environmentBlendMode must not be opaque.[Imm18b].

Using navigator.xr.supportsSessionMode(), it is possible to check if desire mode is available for the device or not. Once desired mode is confirmed, the application can request an XRSession through navigator.xr.requestSession(), so that it can interact with the hardware of XR device.

When XRSession starts, to provide the content to the users, some setup need to be done.

In first place, API needs to create XRReferenceSpace, which is responsible for establishing a 3D space, in which XRViewerPose data will be defined.

Secondly, it creates an XRLayer which is set as base layer of render state. It is defined as XRSession's renderState.baselayer. It is set as base layer because at this moment, API only support one layer but in future, API will start to support multi-layers, where other layers will be able to put on the base layer.

Thirdly, to show the content continuously to the users, API needs to start the render loop pumping. It can be started by calling XRSession.requestAnimationFrame.

XR Frame

An XRFrame provides all the information about the state of all of the tracked object for that particular frame for an XR session. Application can access XRFrame through requestAnimationFrame() method with XRFrameRequestCallback. *Animation Frames* has the information about the tracking state of the XR device, and XR session has a list of animation frame callback which is associated by animation frame callback identifier. At starting animation frame callback identifier is zero and list of animation frame callbacks is empty. With each frame, identifier number is increased by one and callback is appended to session's animation frame callbacks list associated with the current identifier number. When animation Frame is called, it provides all information of device of

that frame in a form of XRFrame.

XRSpace and XRReferenceSpace

XRSpace is responsible to provide an entity which is tracked by the tracking system of the XR devices. XRSpaces may have different three Dimensional relationship to one another in different frame, but it can be evaluated through getTransformTo() method in ever every XR animation frame.

When XRSession is created, we need to map a 3D world, which should be fixed through the entire session, so that each virtual object which is placed in a position of the real world, stay in same position all over that session. *XRReferenceSpace* is responsible for it. XRReferenceSpace describes an XRSpace, which is expected to remain fixed entire XRSession.

As we know there are 3 dimensional coordinate system X, Y and Z, for XRReferenceSpace system they are defined as

- X axis represents left or right direction, where positive value considered as 'right'.
- Y axis is aligned with gravity which represents up or down direction, where positive value is considered as 'Up'.
- Z axis represents forward or backward direction, where negative value considered as 'Forward'.

XRReferenceSpace can be categorized with three types: XRStationaryReferenceSpace, XRBoundedReferenceSpace, XRUnboundedReferenceSpace

1. **XRStationaryReferenceSpace:** It can be initialized by enum value 'stationary'. This mode is effective, when user does not move around the 3D space. Even if the XRDevice has capabilities of 6DOF tracking system, it does not provide information of position of the user in 3D space. When the user moves, the distance between object and user remain same.

   There are three subtypes of XRStationaryReferenceSpace.

   - Eye-level: During creation of an XRStationaryReferenceSpace, if we set subtype as eye-level, It will create an XRStationaryReferenceSpace with it's origin positioned near the eye-level or head level of the users. Some use-case of this types are Airplane simulator, Rolar-coster 3D Video viewer, Solar system explorer, car racing game etc.

- Floor-level: If floor-level is set as a subtype, it will create an XRStationaryReferenceSpace with it's origin positioned at the floor where the user is standing during the time of creation. Therefore, the value of y axis is equal to zero, though the value of x and z is initialized based on conventions of the underlying platform.
- Position-disabled: It can be set as subtype during XRStationaryReferenceSpace vreation, when orientation is tracked but the position of viewer is reported as being at the origin [Imm18b]. This subtype is mainly useful to present pre-rendered media like panaromic photots or videos. This type should not be used for other types of media.

2. **XRBoundedReferenceSpace:** In this category, it creates a floor-relative tracking space, where boundary of the space is pre-established and users can freely move within this boundary but not expected to move beyond this pre-defined boundary. As it is floor-relative tracking space, so the origin of XRBoundedReferenceSpace should be positioned at the floor. Therefore, y axis should be 0 at floor level and the x and z position and orientation are initialized depends on the conventions of the underlying platform[Imm18b] and w value of each point must be 1 to maintain the perspective. Normally it is better to position them near the center of the room facing in a logical forward direction for better user experience. Some use cases of XRBoudReferenceSpace can be used are VR painting/sculpting tool, Training simulators, Dance games, Previewing of 3D objects in the real world.[Imm18b]

3. **XRUnboundedReferenceSpace:** In this category, it creates a tracking space, where users can move freely around the environment, even users can go far from their starting point. If the users move very far from their stating point, the origin may shifted to its original physical location and it will be initialized at near the user's head during creation.[Imm18b]

XRViews

XRView represents a single view into an XR scene. Our end goal is to create a believable view to the user, so that user feels that all virtual objects exist in real environment or entire virtual world is real. To present imagery to the user, each view corresponds to a display or portion of a display of the XR device [Imm18b]. During presentation of the scene, different devices may request different number of views. As example many HMDs request two views (rendering of the content) for the better perspective view for the users. One view is for left eye, other view is for right eye. When an object is near to the user, depending on position of eyes, position of the object seems differently respect to its background, therefore, the view for left eye and right eye can be different. In other hand, many magic window devices request only one view. As the application should be able to run in any device, so it should not assume the specific configuration of the view. XR View provides all necessary information to render content according to the view's

physical output properties such as field of view, eye offset etc.

XR Views interface has four readonly attributes

- eye: To request the view for specific eye, the eye attribute can be used. It has two enum values: left and right. Eye attribute ensures that right content is rendering according to the position of the specific eye and displaying the view to the correct eye. But if application needs to request just one view, does not need eye specific view, attribute should be set as 'left'.
- projectionmatrix: This attribute is responsible for providing a 4X4 matrix according to the projection of the device. It can be called as WebXR Matrix. The 4X4 WebXR matrices have 16 float32Array elements in column major order. API needs this matrices during rendering the view to be imagery to the users.
- viewMatrix: During rendering the view, API needs view transform, which is provided by viewMatrix. ViewMatrix is also a 4x4 webxr matrix.
- transform: The users always change the position and orientation of the device to experience augmented reality or virtual reality. The transform attribute provides all the information about this transformation of the device.

XRViewPort

During rendering the content for the view, API also needs to know a rectangular region of a graphics surface to be rendered. The rectangular region is called viewport. XRViewport provides the information about a viewport of a graphics surface[Imm18b]. It has four attributes x,y, width and height, where x,y defines a position which is the offset from the surface origin and the rectangular dimensions of the viewport is provided by width and height.

XRRigidTransform

It is a transform, which provides the position and orientation of the device. The transform attribute of XRViews use XRRigidTransform to render the XRView.
XRRigidTransform has three attributes: position, orientation and matrix, where both position and orientation are DOMPointReadOnly (W3C) attribute and matrix is Float32Array.

The attribute *position* contents 4 values: x, y, z and w, where x, y and z is 3- dimensional point, which is given in meters and the value of w for position must be 1.0 (w is scaling factor). If position is not a DOMPointInit, the values of x, y, z and w of position are initialized as x=0.0, y=0.0, z=0.0 and w= 1.0. Then these values are updated according to the respective dictionary number of x, y and z, where the value

of w always should be 1.0.

The attribute *orientation* also contents 4 values same as position and it is initialized as x =0.0, y=0.0, z=0.0 and w= 1.0. Then it is updated according to the device's physical attributes, though the value of w always should be 1.0. during change of orientation device, the value of x, y, z and w are updated according to the respective dictionary number. The value of orientation must be normalized to have a length of 1.0 i.e. the value of x, y and z should be divided by value of w to get the value in a length of 1.0.

Identity transform: when the values of x, y, z and w for both of position and orientation of XRRigidTransform are x=0.0, y=0.0, z=0.0 and w =1.0, it is called identity transform.

XRRay

XRRay represents a origin point and direction of a geometric ray. XRRay has three attributes: origin, direction and matrix, where origin and direction both are DOMPointReadOnly (W3C) attribute and matrix is Float32Array.

The attribute *origin* has 4 values x, y, z and w. If origin is not a DOMPointInit, the values of origin are initialized as x=0.0, y=0.0, z=0.0 and w =1.0. Then these values are updated according to the respective dictionary number where the value of W is always set as 1.0.

In the same way, the attribute *direction* vector has also 4 values x, y, z and w. If direction is not a DOMPointInit, the values of direction are initialized as x=0.0, y=0.0, z=-1.0 and w =0.0. As from 2.7.3, we know that z axis represents forward and backward direction, where negative values represents forward direction, hence the value of z for direction is initially set as -1.0. Then these values are updated according to the respective dictionary number where the value of W must be set as 0.0 and direction vector must be normalized to return the value in a length of 1.0.

XRViewerPose

Till now we have found that how XRRigidTransform and XRRay works in terms of position, orientation, origin and direction. And XRView renders content to fit the scene into XRViewport. But which content should be rendered, it depends on the state of XRDevice. When users move their heads or move the XRdevices to look around or to move around the environment, application needs to track the position, orientation and the direction of the device to the API, so that based on these data XRView can generate the scene. In each XRFrame, scene needs to be drawn from the perspective of a viewer.

XRViewerPose tracks the position and orientation of the viewer relative to the XR-ReferenceSpace [Imm18b]. Here viewer can be described as XR device or an user who ware an XR device on the head to experience AR or VR. XRViewerPose has two attributes: transform and views.

The attribute transform is a XRRigidTransform of the viewer relative to the origin of XRRegerenceSpace.

The attribute views is an array of XRViews which hold the viewpoints of XRscene through viewMatrix and a projectMartirx relative to the XRReferenceSpace that are used during rendering of the content with WebGL.[Imm18b]

XRLayer

XRLayer is the content, which is presented to the users through the XRDevice display. It is set as a base layer of render state. It represents a source of bitmap images and a description of how the image is to be rendered to the XRDevice.[Imm18b]. The current version supports only one layer which is set as base layer, but in future version, it will support to create multiple XRLayer, which can be added on base layer.

There is only one supported layer type of XRLayer, which is XRWebGLLayer, though in future new layer type will be added, which will provide opportunity for developers to use any other new graphics API that is possible to render through browser.

XRWebGLLayer: XRWebGLLayer is a layer which allows WebGL to provide bitmaps to be rendered and enabling hardware accelerated rendering of 3D graphics, to be presented on the XRDevice.[Imm18b]

WebGL Context Compatibility: To use WebGL canvas with an XRWebGLLayer and to use WebGL context as a source XR imagery, the context should be compatible with XR device. There are different type of XR Device with different environment. In desktop computer, XR device will be plugged in. Therefore, to make the context compatible with XRdevices, webGL context should be created against the graphics adapter of the XR Device which is physically plugged in. But mobile device is XRdevice itself, hence WebGL Context compatibility is not a concern. It should be always compatible.Typically, WebGL context should created on compatible graphics adapter before in can be used with XRWebGLLayer.[Imm18a]

XRPresentationContext

As our goal is to enable XR application through web, therefore, we have to present our view through HTML. XRPresentationCotext is a wrapper object for the

domxref("HTMLCanvasElement") object. AR or VR content is being drawn on this object which is visible to the viewer.

Events

In the section 2.7.3:XRHardware, we have described that, during the session, if XRHardware is changed or become unavailable, user agent calls devicechange event, so that we can handle the changes through the steps described in 2.27. Hardware changes is one of the situations. There are many other scenarios that may happen during an XRSession which are needed to be considered and should be handled through different types of event. Before discussing about the events, let's take a look from the different types of scenario from where event should be called.

1. During an XRSession, users may tap on the URL bar to type a new address. In this situation, View of AR or VR should be blurred, though XRSession should remain active but tracking should be prevented, so that user can interact with UI efficiently

2. If the display resumes from the paused state when tracking was prevented, an event should be called, so that tracking system starts again.

3. During an XRSession, users may want to interact with the application such as touch or move an object, controlling an object through voice command etc. Therefore, we need to trigger an event to recognize the sources of input devices.

4. The system may have multiple controllers as input sources to interact with the XR application. The controllers may change or unavailable during an XRSession, which is needed to be detected through an event.

5. During the XRSession, when users give any command to XR application, XRSession should trigger an event so that system can execute the command. And the end of the execution of the command, system should call back an event to confirm that command was executed successfully.

6. The anchor point of the 3d world space, which is identified through XRReferenceSpace may need to be changed, if the system loss the known anchor point or user agent find that boundsGeometry has changed. To handle this, user agent triggers reset event to map the world again with new anchor points.

7. At the end, users may terminate the application, or the app can be terminated by the user agent anytime, which should stop the XRSession.

To handle the scenarios discussed in above, XR App must have different events call.

- **XRSessionEvent:** It is called if the state of XRSession is changed. The events list for XRSessions are

  onBlur: It is used, if the users pause the display for any reason such as url typing. This event also can be called if the display is paused by user agent, operating system or XRHardware. This event can be used for the scenarios discussed in 1

  onfocus: This event is called when display is resumed from its pause state by user agent, XRHardware or operating system. This event can be used for the scenario 2

  onresetpose: This event is called when the pose presentation of the display needs to be reset.

  onend: This event should be called when XRDevice is stopped as described in the scenario 7

  onselect: It indicates that an input devices has activated or released.

  onselectstart: It is called when an input device is activated.

  onselectend: It is called when an input device is released.

- **XRInputSourceEvent:** This event returns information of the controller device that will be used to control XR apps. The action through a controller device can be trigger, touchpad, button, voice command or hand gesture. This event is useful for the scenario that we discussed in 3, 4 and 5. This list of inputsources can be found through an array of XRSession.getInputSources()

- **XRReferenceSpaceEvent:** At detection on the changes of the state of XR-ReferenceSpace (discussed at 2.7.3:XRSpace) results calling the event 'XRReferenceSpaceEvent'.

2.7.4 WebXR Polyfill

WebXR-pollyfill is a codebase that provides some sample code demonstrating the use of WebXR API. This polyfill is not dependent on any external libraries but it provides the class XRExampleBase which can be the starting point for working on WebXR session and rendering into a WebGL layer using Three.js.
Using this framework, apps can be created for ARKit compatible iOS devices, ARCore enabled android devices and as desktop application also. Webxr-pollyfill takes care of the handling of the data based on the platforms, by calling different classes internally. This framework is still unstable so while creating apps you can face some challenges [Moz18].

2.7.5 OpenCV

In augmented reality, Object detection plays an important role. To implement object detection in our application, OpenCV has been used. OpenCV (Open Source Computer

Vision Library) is an open source computer vision and machine learning software library which is released under a BSD license.[Ope18c] Therefore, it can be used for free of cost for both academic and commercial sector. The project was initially lunched by intel in 1999, later it was supported by Willow Garage and Itseez.

OpenCV library is mainly focused on real-time computer vision system. It is widely used for video and image processing. Though OpenCV library was primarily written in C++ but now it has Python, Java and MATLAB interface, which supports all popular operating system i.e. Windows, Linux, MacOSX, iOS and Android. The library has more than 2500 optimized algorithms which can be used for image detection and recognition, object identification, camera movement tracking, tracking of moving objects, extracting 3D models of objects etc. OpenCV library is now available in javascript (OpenCV.js), which has been used to develop XR application for website in this thesis.

OpenCV.js Web is the one of the most computing platforms, which are not only used for website browsing any more. Every browser implemented with HTML5 standard now supports online video rendering, capture webcam / camera video through webRTC API. Even the browsers are able to access each pixel of video frame through canvas API.

OpenCV.js is a javascript library of OpenCV, which allows emerging web application with multimedia processing. It leverages Emscripten to compile OpenCV functions into asm.js or WebAssembly targets and provides a JavaScript APIs for web application to access them [Ope18c]. To develop an augmented reality or virtual reality app for web, developers need a wide array of image and vision processing algorithms in javascript which can be implemented using OpenCV.js.

Haar Cascade in OpenCV: OpenCV comes with haar cascade algorithms implemented which we discussed in 2.6.6, therefore we can easily implement object detection feature in our application using OpenCV. OpenCV also allows us to create our own haar feature classifier.

To create our own haar classifier, we need to provide a set of positive images, which contains the object of interest and a set of negative images, which does not contain the object of interest.

Negative Samples: Negative samples are taken from the set of provided negative images where the object we want to detect is not present. All the negative images should be listed in file (usually a txt or dat file) that contain the path of an image per line. Negative samples are also called background images. Negative images can be in different size but must be equal or larger than desired training window size. Directory structure can be looks like

```
/negativeImage
  negativeImg1.jpg
  negativeImg2.jpg
```

```
  negativeImg3.jpg
negatives.txt
```

And negatives.txt should be look like

```
negativeImage/ negativeImg1.jpg
negativeImage/ negativeImg2.jpg
negativeImage/ negativeImg3.jpg
```

If all negative images are in same folder, we can create this file very easily using command line

```
find ./negative_images -iname "*.jpg" > negatives.txt
```

Positive Samples: Positive samples are created from the positive image through opencv_createsamples application. It is possible to create positive sample data set in two ways. - A bunch of positive samples can be generated from a single positive object image. - We can provide all the positive images, in this case this toll will be used only to cut them, resize and convert them in opencv readable binary format. We should provide different types of positive image, where object appears in different way, different angle and perspective. To run opencv_createsamples, we can set different parameters. Some of the most useful arguments are

```
-vec: Output files that contain the positive samples for training
-img: Source object image.
-bg: Background/negative description file
-num: The number of positive samples, we want to generate.
-bgcolor: Background color (grayscale images are expected).
          It also denotes the transparent color.
-bgthresh: The threshold for background color
       (the amount of color tolerance)
-inv: To invert the colors
-randinv : To invert color randomly.
-maxidev: Maximal intensity deviation of pixels in foreground
          samples.
-maxxangle: Maximal rotation angle towards x-axis
            (value should be in radians)
-maxyangle: Maximal rotation angle towards y-axis
            (value should be in radians)
-maxzangle: Maximal rotation angle towards z-axis
            (value should be in radians)
-w: The width of the output samples in pixel.
-h: The height of the output samples in pixel
```

*source [Ope18a]

In next step, opencv_traincascade command is use with different arguments for cascade training. Some of the common arguments are

```
-data: The directory name where the trained classifier should
be stored. This directory also stores the output of each stage
as xml file.
-vec: The vec file name that contain positive samples which are
created by opencv_createsamples
-bg: Background / negative samples description files (example:
     negatives.txt)
-numStages: The number of cascade stages that we would like to
train to get our classifier.
-minHitRate: Minimal desired hit rate for every stage
-maxFalseAlarmRate: Maximal desired false alarm rate for every
                    stage
-numPos: The number of positive samples that will be used in
every stage for training
-numNeg: The number of negative samples that will be used
in every stage for training
-w: The width training samples in pixel. It must be the same
value that we used in create_opencvsamples -w
-h: The width training samples in pixel. It must be the same
value that we used in create_opencvsamples -h
-mode: Three types of mode can be used for training i.e. basic,
core, all. Default value is basic.
-precalcValBufSize: Size of buffer for precalculated feature
values in mb. High value speed up training. But sum of
precalcValBufSize and precalcIdxBufSize should less than memory
size of the computer.
-precalcIdxBufSize: Size of buffer for precalculated feature
indices in Mb. High value speed up training. But sum of
precalcValBufSize and precalcIdxBufSize should less than memory
size of the computer.
```

*source [Ope18a]

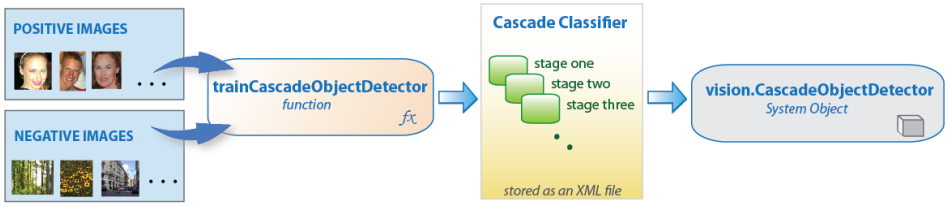

Figure 2.28: Steps of cascade classifier [Ber]

During training cascade, after completion of each stage, the output of training is stored in a xml file as shown in figure 2.28. At end of training of all stage (the values

we provided for numStages), the final cascade classifier is created. Figure 2.29 shows the overview of cascade classifier training.

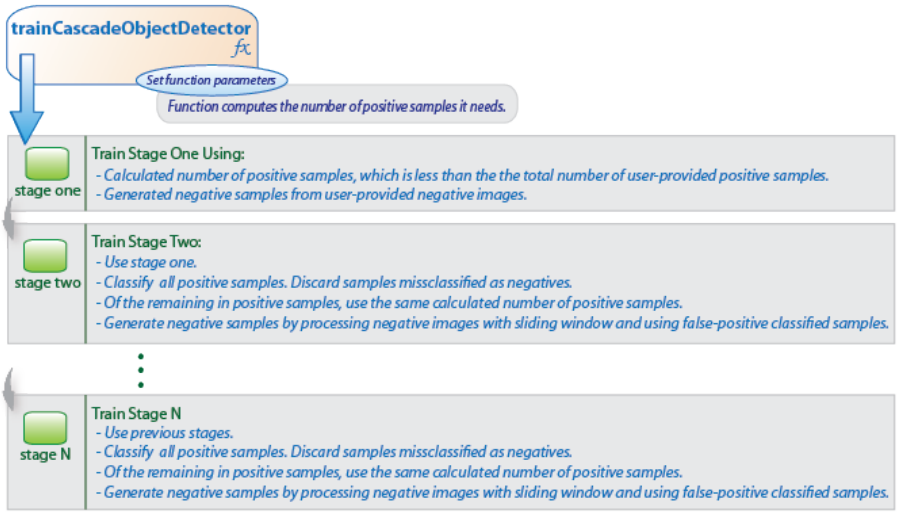

Figure 2.29: Cascade classifier training overview[Ber]

2.7.6 TensorFlow

To detect and recognise object using *convolutional neural networks*, we have used tensorflow api. TensorFlow is a fast and flexible open-source machine learning library to develop and train ML models for research and production. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow was developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for their internal uses. [Ten19b]

TensorFlow Object Detection API:

The TensorFlow Object Detection API is an open source framework, which is built on the top of TensorFlow. This api is mainly used in computer vision for object detection, recognition and localization (where is the object in the scene). It allows us to easily construct, train and deploy object detection models.

COCO-SSD:

It is an open source library, which is also available for nodejs. This library is focused on localization and identification of multiple objects in one frame. It is built on the top of the TensorFlow's Object Detection API. This library is capable to detect 90 classes of objects which is defined in COCO dataset. SSD is the abbreviation of Single

Shot Multibox Detector. [Ten19a] Single Shot Detector method gained a good balance between accuracy and speed.

2.7.7 3D Library

ThreeJS

ThreeJS is a java script library and API which helps in creating and displaying animated 3D objects in web browser. It was first released in April 2010 [Thr18a]. ThreeJS allows developers to create GPU-accelerated 3D animations with the help of JavaScript and it does not depend on the browser plugins. We have used this library to create the virtual objects in our app, like the tv-remote, the board and the gems

TweenJS

It is a javascript library used for twining and animating properties of javascript and HTML5. In twining process intermediate frames are generated between two images so that the transition between images looks smooth [Thr18c]. TweenJS works within ThreeJS render loop and it enhances the overall performance of the application and also helps in keeping a high frame rate. We have used TweenJS for the smooth movement of the gems [Thr18b].

2.7.8 NodeJS

NodeJS is an open-source, cross-platform run-time environment which is used to execute JavaScript code outside the browser. Using NodeJS user can write server-side scripting. Node.js was originally written by Ryan Dahl in 2009[Nod18b][Nod18a]. NodeJS is an open source server environment and it is free. It uses JavaScript on the server. WebXR-polyfill uses NodeJS as the server.

2.7.9 Webpack

To run XR application, we need to load all different JavaScript files, api (such as webxr-api, TensorFlow api, TensorFlow-Object Detection api etc.) into the browser. Webpack is a static module bundler which is mainly used to bundle all JavaScript files for usage in a browser. During webpack build, it process all dependency files and generates one or more bundles. It is also capable of transforming, bundling, or packaging just about any resource or asset [Web19].

2.7.10 Samsung TV API

Smart TV is getting popularity now-a-days, which have access to the internet that allows users to browse website or to install different application. Therefore, the smart

tv manufacturers provide different type of SDK packages, libraries and API, so that third party developers can develop application for the smart television. In our thesis, we choose Samsung television to provide demo.

Samsung provides API in different languages for the developers to establish communication with Samsung Television from the application. One of them is JavaScript API which allows developers to send and receive data from the web-based application[Sam18]. Based on this API, many Samsung smart tv remote packages have been developed from third party developer which support nodejs. In this thesis, we have used node-samsung-remote npm package [bee17] to establish communication between our XR application and Samsung television.

2.7.11 RobotJS

In our thesis, one of our use-case is to create a laptop screen act like touch screen. Which means, through our XR application screen, user should be able to touch or click any button on the laptop, that s/he can see through the camera. To implement such a feature, we have taken help from RobotJS library.

RobotJS is a desktop Automation Library for node.js which supports MAC, Windows and Linux. Using this API, it is possible to develop an application to control desktop from remote through mouse and keyboard. Such as using this api developers can get the position of mouse, can change the mouse position, can send the keyboard command, get screen resolution, get pixel color of mouse position etc.. As this library is still in progress, therefore it also has some limitation, such as it does not support multi-monitor completely. Though it returns current mouse position, even if the position of the mouse is in secondary monitor (It can be negative value) and using this returned value, limited control of mouse is possible. The other features that are not supported by this library are global hotkeys, screenshot through robotJS.[Rob18]

RobotJS can be easily installed by the command with

```
npm install robotjs
```

Python v2.7.3 is recommended as currently v3.x.x is not supported by robotjs Using RobotJS, other developers have developed some cool apps such as screenCat, LeapGamePad, WebbyMouse, PixelColor, Remote Control Server.

In our thesis, to control mouse and mouse click through touch on mobile screen, we have used Remote Control Server developed by [jer18]. Remote Control Server allows users to control desktop from any web browser on other computer or mobile. It is developed on the top of robotjs. We have modified this open source code according to the requirements in our XR application.

3 Use-Cases and Requirements

We have considered different use-cases focusing on the problems that are discussed in the problem statement section 1.2. For experiment purpose, we have developed different XR applications for each use-case scenario. In this section, we will explain the use-cases along with concept and features of the application for each use-case.

3.1 Use-Cases

3.1.1 Object detection: Detect and control device through Web-based XR application with OpenCV

Like 'plug and play' feature, we can use object detection functionality as 'detect and control'. In our first use-case, we have developed an application which is able to detect a device and the user can control the device through the application from the mobile device. In this example, we choose TV/monitor as our target device.

The problem with detecting such a device is, in object detection, we can not separate the television and monitor from each other as both of them look same. So we have implemented two different features so that our application can be useful for both television and computer.

The features of the application as

- If the application detects the object as television/ monitor, it should inform users that tv/monitor is found, and application should be able to connect with the television/ computer using any network protocol such as wifi.
- On detection of the device, a virtual remote will be created to control the TV, such as volume control, channel control, etc.
- In case of television, users should be able to control the device through the virtual remote.
- In case of a computer, the mobile screen can be used as touch screen for the computer i.e., user can click or open any folder just by touching on the folder that appears on mobile screen through the device camera.

3.1.2 Object detection: Web XR implementation inside a regular e-commerce website

We have discussed in problem statement [1.2] that, to develop native AR/VR application from each product seller is not feasible, rather it suits more with web distribution. To experiment with this feasibility of AR features in e-commerce website, in this use-case, we have developed an e-commerce website for sunglass store. Using this application users should be able to give a trial of the sunglass using face detection and easily should be able to change the sunglass to try different on run time.

Features of this application,

- User can choose any sunglass from the product list to give a trial of the sunglass.
- When user will choose one sunglass, the application will detect the face and put the sunglass on the user's face
- Users easily should be able to change the sunglass to try different sunglass on run time from the list of the sunglass at the bottom.
- Users can browse the website through mobile phone, tablet or computer. Computer does not have many AR device component such as gyroscope or accelerometer. As in this case, we only need to recognize the face, therefore application does not need to access other XR hardware. Hence the application should work both in computer browser and mobile browser.

3.1.3 Multiple objects detection using TensorFlow

In our previous use-cases, we have implemented object detection technique using OpenCV. Object detection using convolutional neural networks is another technique which has gained popularity on recent time. Therefore, we have implemented TensorFlow's object detection API with webXR API. The main objective for developing this application is to compare the performance with the application implemented with OpenCV, though we have developed this app for different usages.

In this use-case, we have developed an application, which is able to detect and recognize multiple objects in the screen and on the tap of the object name, users are able to learn more about the object. This implementation is not only useful for gaining knowledge. It also can be used for different purposes, such as in e-commerce website, by detecting an object, it can provide the information with prices of the similar products from their store. In this case, providers must train the models of all the products, which are available in their inventory list.

The feature of this application:

- The application will be able to detect multiple objects on the screen.

- Every object should be shown to the user through the bounding box with the name of the object.
- Once user will tap on the name of any object, application should be redirected to the another page with more information about the detected object.

3.1.4 Web-based XR Board game

The goal of this use-case is to develop a web-based AR game. Here, we have implemented augmented reality with a virtual board game. This board game can be played using mobile or tablet. This game is the simplified version of childhood game which use to be played between two persons. We have designed the game in such a way so that the player will be able to see a virtual board inside the real room or environment which is basically captured by device camera. The player can see the board from any direction by placing the camera in different position. In this 3D board game, player will play against the computer. The player can give his/her turn by touching the gems. Based on the turn of the player, computer will play and give its turn. AI (Artificial Intelligence) has been implemented to take care of computer's turn against player. At the end, computer or player will win the game according to the game rules. In the next subsection, we have described the rule and scoring system of the game.

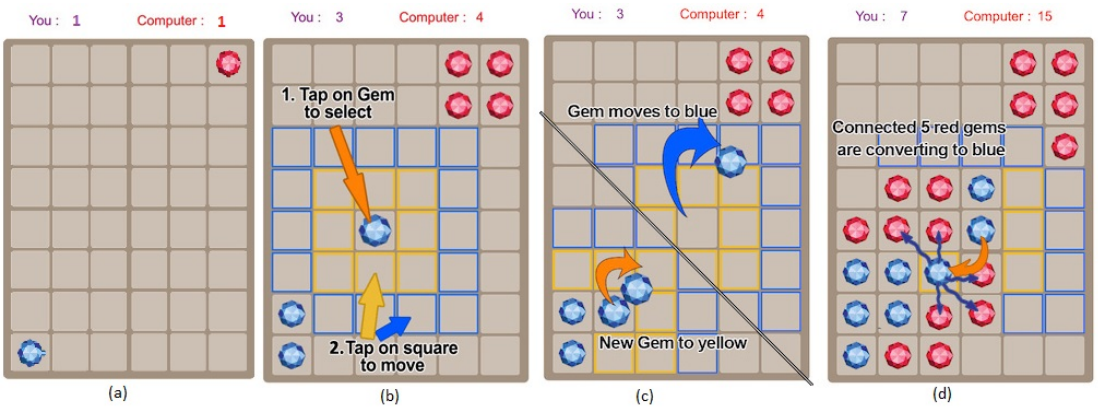

Figure 3.1: (a) Starting point (b) How to move gems, (c) Gem movement rule, (d) Gems color changing

Game Rule

1. At the starting, both players will have same amount of gems as shown in Fig 3.1(a). Blue gems are for the player and red gems are for Computer.

2. Gems can move up to 2 available adjacent blocks. In Fig 3.1 (b), we can see the available blocks where the selected gem is allowed to move.

3. If we select the neighbor block (yellow highlighted), it will result into a new Gem creation. On the other hand, selecting 2nd neighbor block (blue highlighted), the existing Gem will move to the new block as shown in Fig 3.1 (c).

4. Once Gem will reach the destination block, it will check if there are any gems of the opponent (here computer) in neighbor blocks, those will be converted to the player's gems. (shown in Fig 3.1 (d)). The same logic has been implemented for Computer's turn as well.

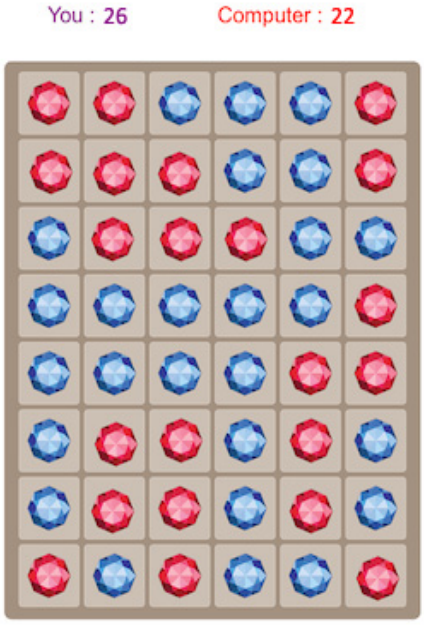

Figure 3.2: Game result: You win

Score Calculation

At the end of the game there should be a result. The result will be decided depends on the score. After each turn, score will be calculated based on the number of gems present on the virtual board for each player.

In Fig 3.1 (a), at starting we can see both of team have same score as 1 vs 1. In Fig 3.1 (b), we can see, that the player (here You) scored 3, while the computer scored 4. In Fig 3.1 (d), the score just before giving player's turn is shown, so player's score is 7 and computer's score is 15. But when that turn will be finished, 6 red gems will be

turned to blue gems and the player's score will increase to 13 and computer's score will be 10. At the end of the game who will have the maximum numbers of gems will be the winner. In Fig 3.2. we can see that player (You/blue gems) holds 26 gems, where computer holds 22 gems(red), so here, player (You) is the winner.

3.1.5 Education through 3D models visualization

WebXR application can be highly used in the education system to explain anything visually to the audience. One use-case is, in medical class, a virtual human 3D model can be presented in a real environment, where the part of the human body is interactable. Student can easily touch on any part of the body to see the information about that part. Not only in medical class, this technology can be used even in engineering class to explain any car or part of engine. The benefit in such kind of application distribution through web is, developers can easily introduce new model just by providing a new link. And students do not need to install new application for each model. They just can search it on the web and can browse any link to learn from a model.

In this scope, we have developed an XR application to show how an app can be easily distributed as web application to provide an AR based visualization of 3D models.

- We have created a web-page, which contains the list of models.
- Developers can easily add any new model into this page by adding the information in the database (json in this project) and by creating a new folder of their application.
- We have created two apps for two different models for demonstration. One model explains about human body and others only focus on the skull.
- Users can browse any model by clicking on the link.
- After launching the page of the selected model, users can select a location where the model should be created.
- Users can change the location of the model using the reset button.
- Once the model is loaded, users are able to view it from any direction by moving the mobile and will be able to touch on any part of the model to know details about it.

3.1.6 Web based XR Game with Object Detection

We have implemented a very basic game to evaluate the performance of Object Detection in a web based XR game. As from the evaluation 6.4 we have found that openCV performed well with less powerful device compared to TensorFlow, therefore we have used openCV for object detection in this game.

As same as the previous game, this game can be played using mobile or tablet. In this game, on a virtual 3D board, a ball moves around the board according to the hit by the player. Players have to touch the ball with the bat to change it's moving direction before the ball touch the baseline of the player. Every time player will be able to touch the ball, s/he will receive 1 point. If a player misses the ball, s/he will lose one life. After every 10 points, the speed of the ball will be increased. To win the game, player have to make 30 points. 3 lives are set at starting of the game. If the player looses 3 lives before making 30 points, s/he will lose the game. Player will be able to move the bat right or left by moving the open-palm in front of the camera. App will detect the palm of the player, and will set the position of the bat according to the palm.

3.2 Requirements

To run webXR application in the browser, we need XR hardware enabled devices. Nowadays many mobiles and tablets support AR functionalities. But not all devices have strong enough processor or graphics card to run all kind of XR applications. Moreover, our application runs on the top of web browsers. So app needs to access the hardware information through the web browsers. But not all web browsers provide AR supports. Here we have described technical requirements that are used to develop the XR applications and also described the supported devices and browsers to run the application.

- **Web Server:** To develop a web application, a web server is needed. We have used nodejs as server-side framework.

- **Accessing Hardware Information:** To develop an XR application for web, we need to access hardware information such as accelerometer, gyroscope, device position. Therefore, we need an API which is integrated with browser and can be accessed the information using JavaScript from webapp. WebXR-device-API is the solution for it. Therefore, our applications are able to run on those browsers only, which support webXR-device-API.

- **Framework for WebXR API**. We have used webXR-polyfill framework developed by Mozilla, where webXR-device-API is structured in such a way that from webXR application any information can be accessed easily.
- **3D library** We have used threejs library to create 3D virtual objects.
- **Version Control:** The internal tub gitlab is used for Version control.
- **Hosting Server:** We have hosted our XR applications in heroku server, as it is free to use and also supports https. Secured http is required to access camera from any browser.

- **Bundle creation:** WebXR-device-API and other libraries needs to be loaded and run on the client's browser so that it can provide all required information to the app. Webpack is used to create one or more bundle using multiple dependent JavaScript files.
- **Object Detection Libraries**: For object detection we have used JavaScript version of OpenCV, which supports Haarcascade object detection methods. We also have implemented our object detection technique in our XR application using convolutional neural network. For this we have used COCO-SSD, which is developed top on TensorFlow API.
- **Communication with TV:** To connect our application with Television and control it, we have used Samsung TV API.
- **Communication with Computer** We have used robotjs API to control the computer/ laptop (click, mouse move etc.) from our web application.

- **Supported Device and browser**
  Mobile device should have ARKit (iOS) or ARCore (Android) support to run the webXR Application.
  - XR Viewer (iOS 11 or later)
  - WebARonARCore (Android)

  Note: The application "e-commerce" and "Multiple objects detection using tensorflow" does not use accelarometer or gyroscope, therefore those applications run in other devices such as laptop browser, Safari (iPhone 7 or later).

4 Concept

In our experiment, we have created a web page, which contains a list of different webXR applications for different use-cases. Any webXR application can be added in this site by providing application folder and some information about the app such as app name, brief details etc. into the database. We have used json file to save such data (sd_applist.json).

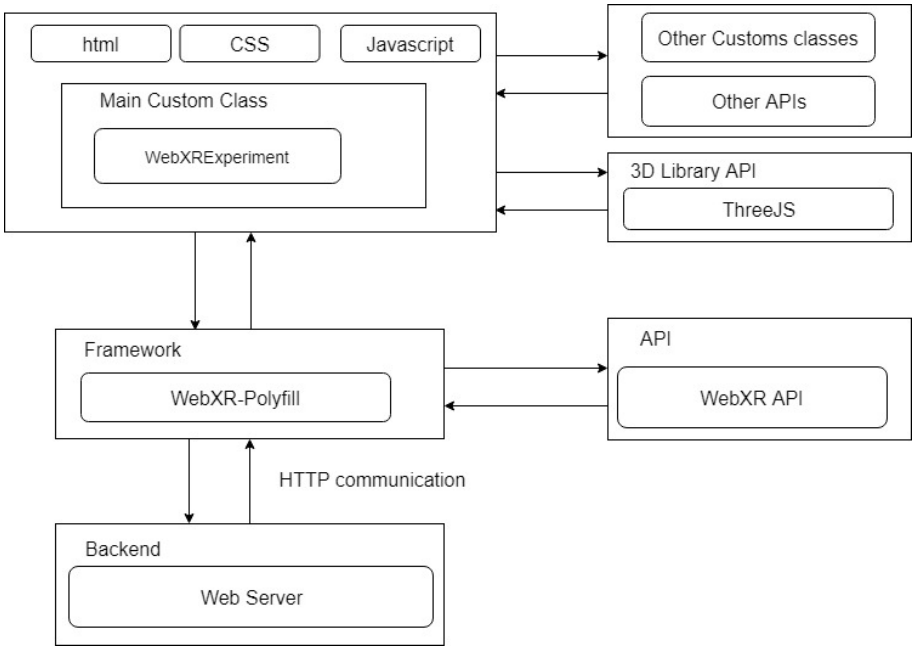

Figure 4.1: Base architecture of the applications

4.1 Common Architecture

The applications have been developed based on three main components i.e., frontend, backend (server side) and the frameworks. We have used NodeJS as web server. The frontend is designed using html, css and JavaScript.

To run webXR application, it requires to access data of different hardware of the client's device such as camera, accelerometer, gyroscope etc. In order to do so, webXR-device-API needs to be loaded into the client's browser during the launch of the application. Therefore, using webpack, webXR-pollyfill framework [Moz18], creates bundles from all dependent JavaScript files of webXR-device-API, which can be loaded into the

client's browser, and webXR application can access any hardware information through this bundles.

WebXRExperiment is the main class for our applications which is developed extending the XRExamplebase class of webXR-polyfill framework. At starting of the apps, WebXRExperiment creates an XR session through WebXR-device-API. This main class interacts with other custom classes, third party api and libraries for different functionalities and requirements.

Each application are developed based on same architecture (fig: 4.1). But because of special requirements for different applications, other components have been added on the top of this common architecture. In our different XR applications, the main class WebXRExperiment is developed in different ways according to the features of the application.

4.2 Extended Architecture for each Application

In this subsection, we have explained the application specific architecture, which is developed on the top of the common architecture.

4.2.1 Architecture for "Object Detection: Detect & Control device"

In this application, we have used OpenCV's Haar cascade classifier for object detection. OpenCV is a heavy weight library, which may take time to load. Another problem is object detection algorithm needs strong processing unit. When application will run, openCV should be loaded and should detect the object without affecting the main application loop. Therefore, we need to consider asynchronous operations, which can be parallelized and able to execute continual running background operations. We can achieve this using the thread operation.

Node.js provides "worker" methods to deal with heavy CPU intensive computations through JavaScript. WebXRExperiment class create another thread using worker-tv.js. worker-TV load openCV and the trained classifier of the object. It requests the current video frame from webXR-device-API, analysis the video frame using haar cascade algorithms and return the output to the WebXRExperiment class with the parameters including number of the detected objects and the location in the frame. Then it ask for the next video frame for analysis. Once the main class (WebXRExperiment) receives the data, it shows if the object is found or not. If the object (TV/monitor)is found, we have considered two different cases for television and monitor. At first let us cosidered that, the detect object is a television. The figure 4.2 shows the architecture of this application.

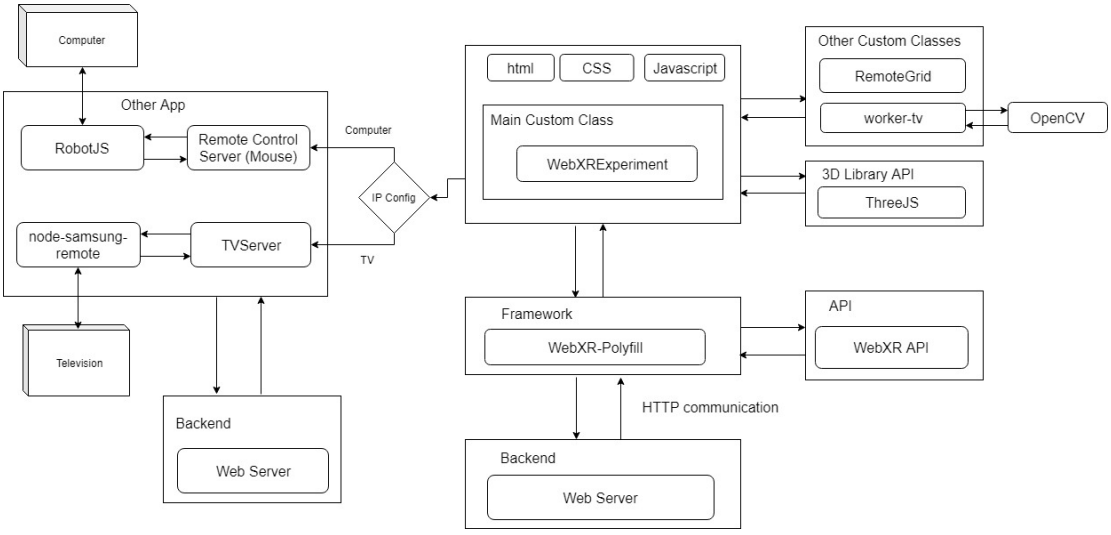

Figure 4.2: Architecture of the 'detect and control' app

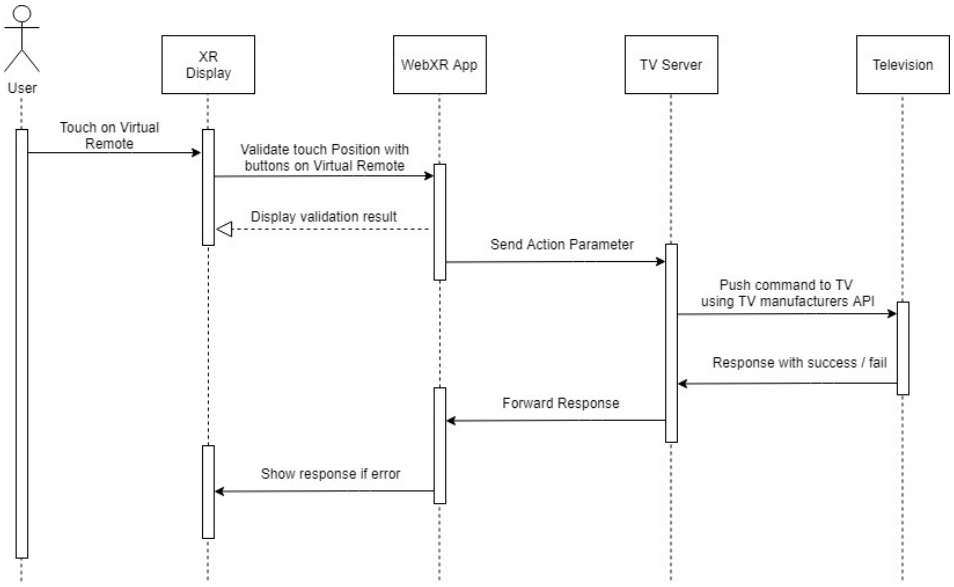

Figure 4.3: UML Sequence to control TV

In case of television, If television is detected, the app asks for the input from the users, where to create Virtual remote. When user touch on the display, the app creates a virtual 3D remote on the touched location in the 3D space. Now user can control the television using this 3D virtual remote. When user taps on the remote button through the display of the mobile, the app validate the touch position with all button to check, which button has been pressed. If the touch location matches with button position, it sends the request with parameters to TVserver, which is running on another web server.

The parameters contain the name of action such as such as volumeUp, volumeDown, channelUp, channelDown, mute etc. TVserver must be configured with television's IP address. TVserver processes the request and send the command to the television using the TV manufacturer's JavaScript API. Within a time limit, it sends back response with parameters (success or fail) to the WebXRExperiment. The UML sequence of this process has been shown in the figure 4.3

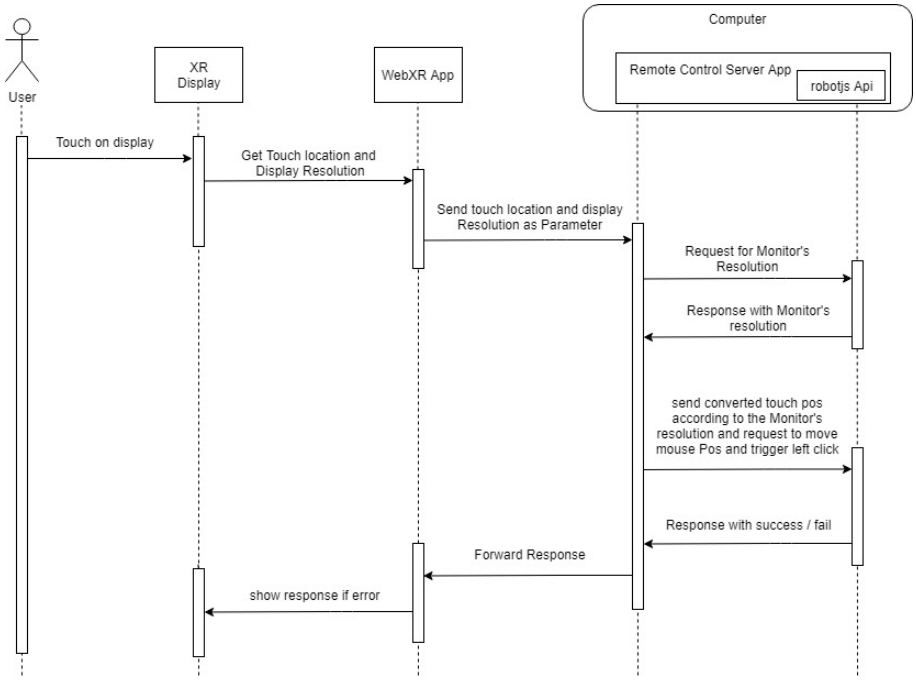

Figure 4.4: UML Sequence to control PC

In case of computer, user can use the mobile display as touch screen. When user touch on any location on the screen, WebXRExperiment take the touch input from the mobile and sends request to remote-control-server with parameters including touch location, mobile screen resolution and action (such as left click). Remote-control-server should be run on the computer which will be controlled. Remote-control-server convert the touch location of the mobile to the respective location according to the resolution of the monitor. Then it moves the mouse into respective touch location and left button click action is triggered. The UML sequence of this process has been shown in the figure 4.4

4.2.2 Architecture for "Object Detection: E-commerce"

As same as previous application, in this application OpenCV has been used for face detection. The webXR application has been loaded upon the request from regular e-commerce website. Once user click or tap on the "Try" button for any product, it sends

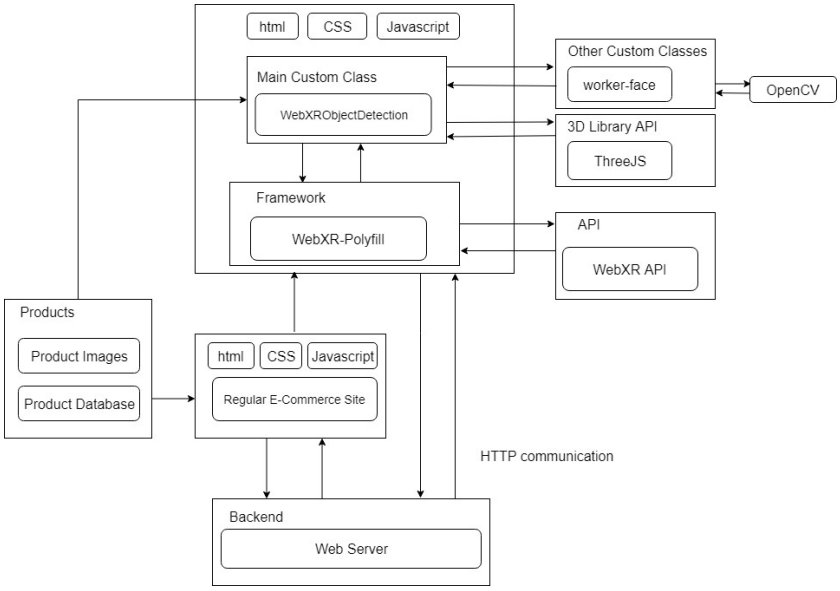

Figure 4.5: Architecture of the E-commerce website with WebXR API

request to XR app with productID. WebXR App detect the face through parallel thread work-face. Once worker-face detect the face, it sends the location and size of the face to the main class. The main class draws the selected product image on the other layer of html canvas according to the face size and position. Figure 4.5 shows the architecture of our implementation.

4.2.3 Architecture for "Multiple objects detection"

In this application TesnsorFlow: COCOSSD library has been used for object detection. At starting, it loads all the cocossd trained model. Worker-tensor provides the video frame on request. The main class "WebXRExperiments" converts the video frame to HTML canvas element and sends it to xr-tensor-bundle for analysis. On callback it draws the rectangle for all detected objects with its name. If user touch on the name of the object, it redirected to wikipedia page of respective object to provide more information. Figure 4.6 shows the architecture of this application.

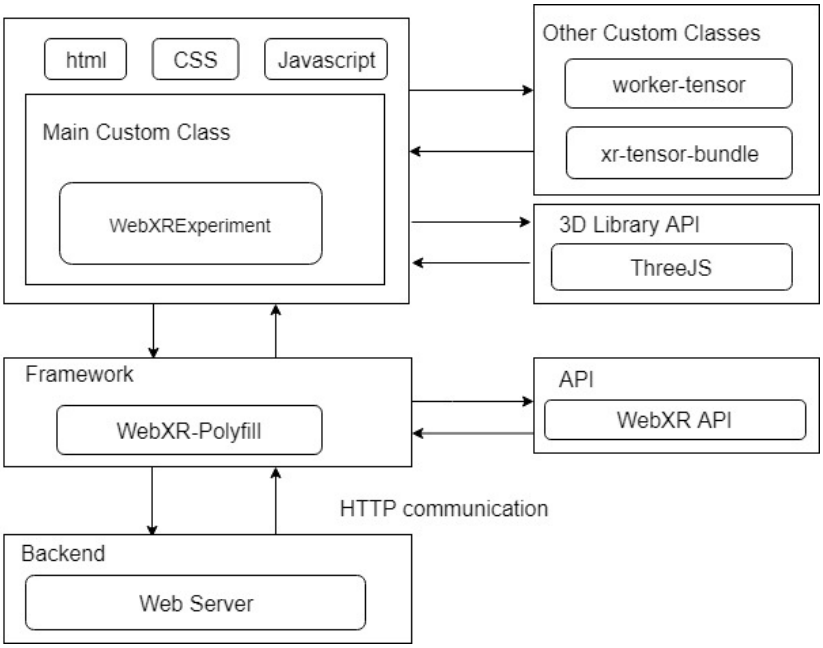

Figure 4.6: Architecture of the XR application with TensorFlow-COCOSSD

4.2.4 Architecture for "Web-based XR Board game"

In this board, we have used threeJS to create 3D virtual elements, tweenjs has been used for animation. Any other 3D library can be used to develop this game. The architecture is shown in the Fig 4.7.

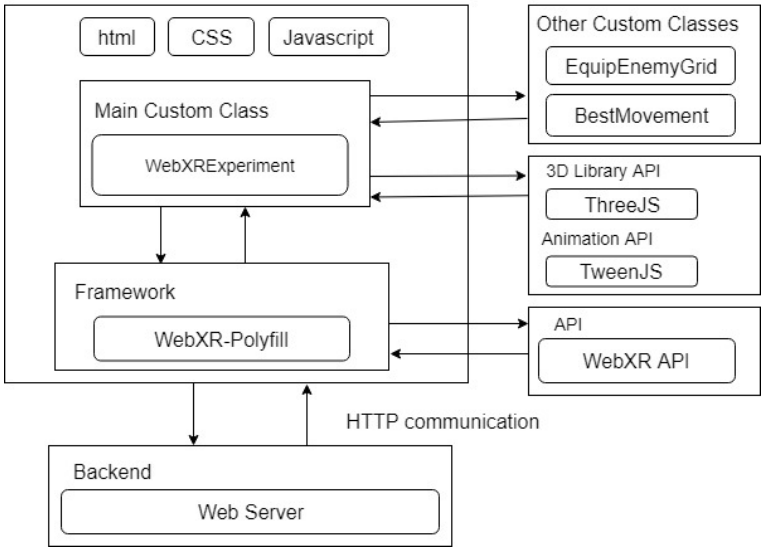

Figure 4.7: Architecture of the WebXR Board game

At loading of the game, if XR session is started successfully, it creates a 3D virtual board and gems using ThreeJS library. Information of each grid of the board and gems are store through the class EquipEnemyGrid. Then the app waits for user's touch to select any blue gem to move it. If user touches blue gems, the app checks the information about neighbor grids from EquipEnemyGrid class and if neighbour grid is empty it shows all the available moves by changing the color of the grid. Then the app waits for user's input. If user touches on any available block, all the gems reload according to the game rule. If the game is not over, it waits for computer's turn. App checks for the best turn using BestMovement Class. Once best movement is calculated, all gems are reloaded again according to game rule. If game is not over, app again waits for user's input. The figure 4.8 shows the flow chart of the game.

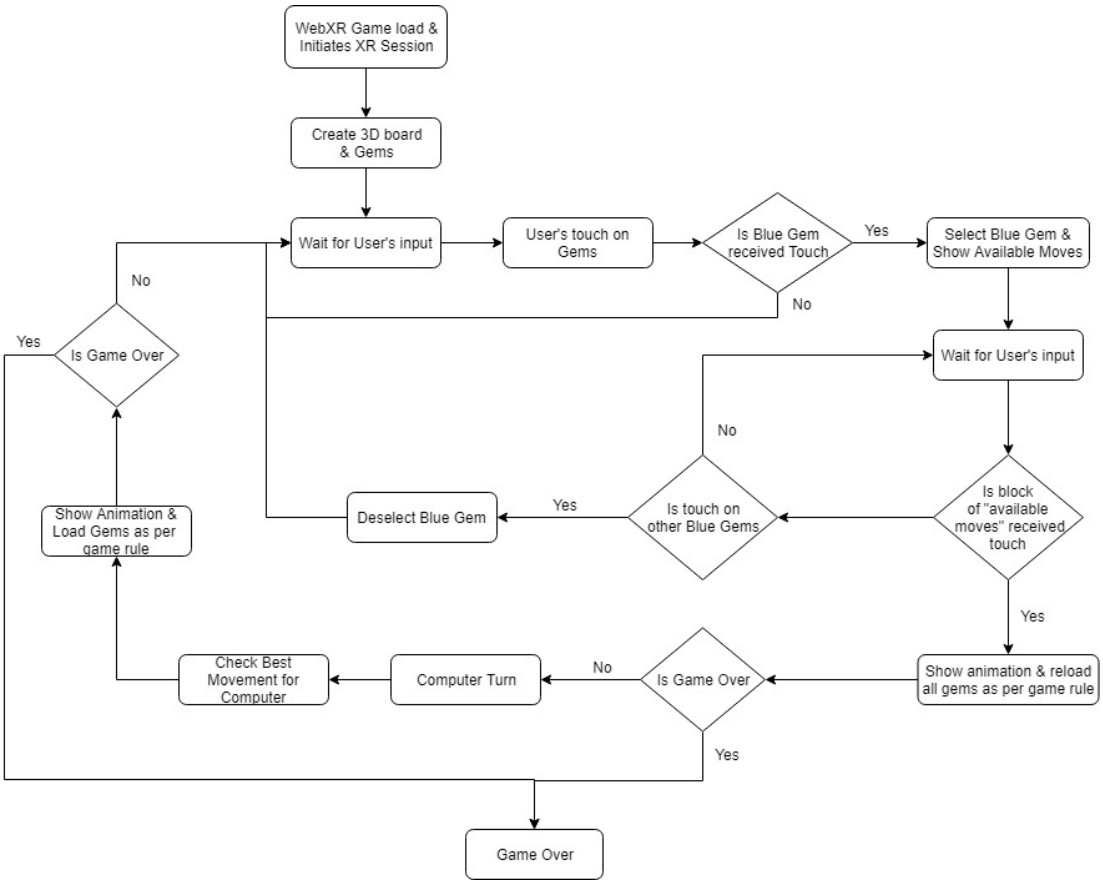

Figure 4.8: Flowchart of the WebXR Board game

4.2.5 Architecture for "Education through 3D models visualization"

In this app, in a page, it shows all the available 3D models from database. Once user selects any 3D models, it loads XR application for the selected model. After XR session initialized, app asks for user input where to load model. When user touch on a position, app calculate the respective anchor point according to the 3D space. If anchor point is found, it creates 3D model using 3D library api. We have used threeJS in our application. After model is loaded, if user touch on any part of the model, the app detect the part which is touched and bend the color with the model. Moreover, it creates a 3D text which shows the name of the body part. The figure 4.9 shows the basic architecture of this solution.

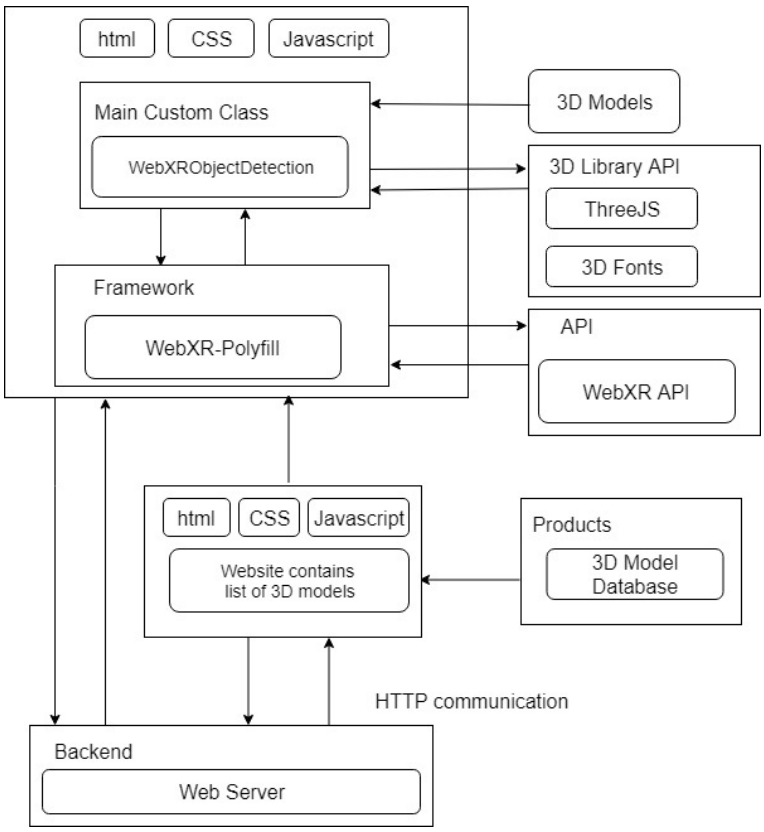

Figure 4.9: Architecture of the XR application for 3D model visualization

4.2.6 Architecture for "Web-based XR game with Object Detection"

The architecture of this game is combination of the architecture of object detection using openCV app and 3D board game. Figure 4.10 shows the architecture of this app.

After loading the application, the creates 3D board, ball and bats. The main custom class WebXRExperiment loads the opencv through worker-palm in other thread. Once openCV is loaded, the app is ready to detect the human palm. To start the game, user need to touch any where on the screen. When user keep his/her open palm in front of the camera, OpenCV detect the palm through worker thread and send response to main class with location of the palm. To keep it simple, we allowed the movement of the bats left and right direction only. Depending on the location X of the palm, we set the location x of the bat. Thus user can move the bat left or right using the open palm.

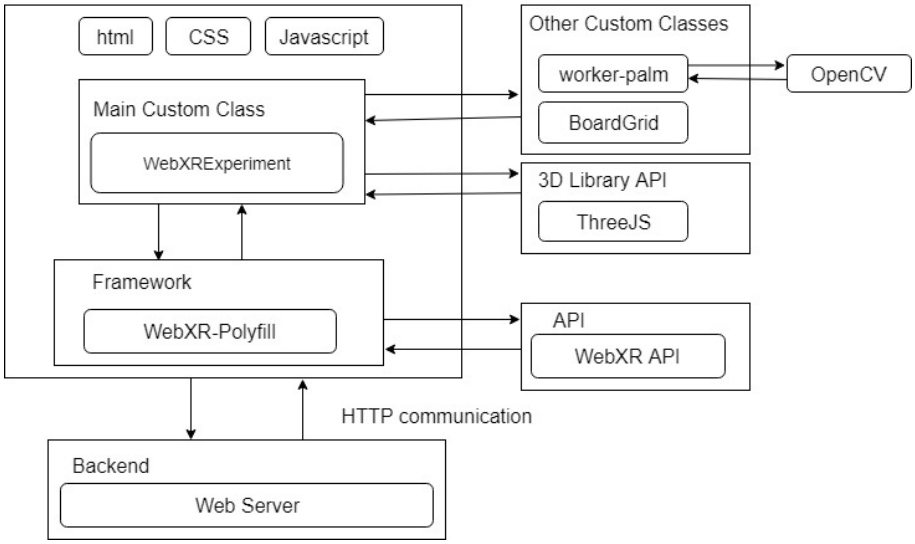

Figure 4.10: Architecture of WebXR game with Object Detection

5 Implementation

This section describes the implementation of our different webXR applications, which are developed using W3C webXR-device-API and other libraries and framework those are described in the chapter2.7. We have developed multiple XR applications for the different use-case scenarios those are defined in the usecases 3

At first take a look from the overall project structure.

5.1 Project Structure

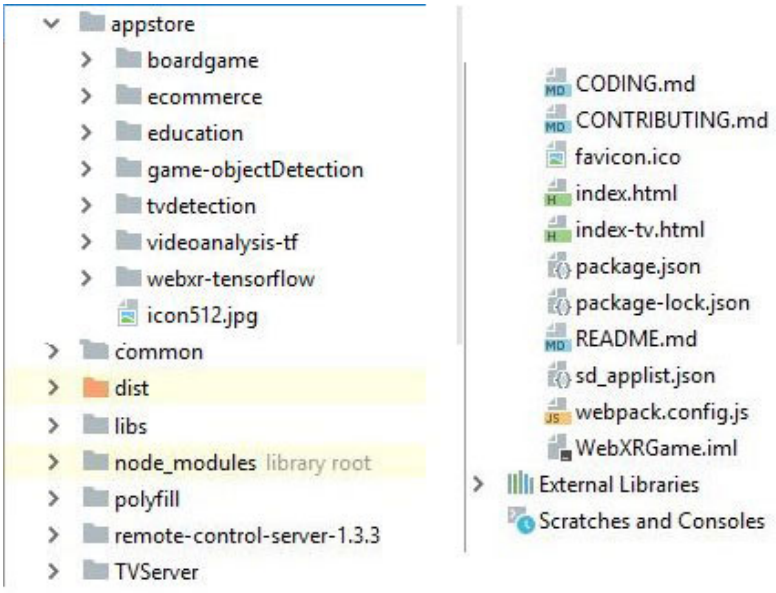

Figure 5.1: Project Structure

Appstore: This folder contains all the XR applications which we have implemented. The sub folders **tvdetection**, **ecommerce**, **webxr-tensorflow**, **boardgame**, **education**, **game-objectDetection** contain the XR application that have been developed for the use-cases object detection: detect and control (3.1.1), object detection: e-commerce (3.1.2), multiple objects detection (3.1.3), webXR board game (3.1.4), education through3D model visualization (3.1.5), and webXR game with object detection (3.1.6) accordingly. **videoanalysis-tf** is the experimental version of use-case 3.1.3,

where we detect the objects from mp4 or mov video instead of webcam. We have discussed in details about the implementation in the next section.

common: This folder contains the files, those are commonly used in all applications such as common.js and bootstap.min.css etc.

dist: This folder contains the bundle files generated by webpack during build with nodejs, which contains webxr-polyfill.js, webxr-worker.js and xrTensor-bundle.js

libs: This folder contains different libraries and framework those are used in our project such as threejs, opencv.js, loader etc.

node_module: This folder is created by the npm install command which contains all libraries and packages including the libraries has mentioned in package.json for nodejs

polyfill: This is the library folder of webXR-device-API which has been structured by mozilla [Moz18] to make it easier for developers to develop an XR application.

5.2 Detect and control device through web-based XR application

This is the implementation of our first use case 3.1.1. In this application our goal is to detect TV/monitor and after detection, user will be able to control it from the mobile device. Figure 5.2 and 5.3 shows the screenshot of this in terms of television and laptop monitor accordingly.

To detect television or monitor, at first we need to create cascade classifier for the TV / monitor using OpenCV haarcascade training that we described in 2.7.5.

5.2.1 Haar Cascade training for TV/Monitor

For the training we have used Ubuntu 16.04 LTE as operating system. Also we would like to thank to Thorsten Ball for the framework[Bal13], which make training process easier and more efficient

Basic installation: Install OPENCV in linux

```
sudo apt-get install python-opencv
```

we need more library for opencv_creatsamples

```
sudo apt install libopencv-dev \\
```

Training: For haar cascade training for the television or monitor, we need to provide positive images and negative images.

Positive Images: We have provided 100 positive images of television and monitor collected from different source using google search. All images were resized with 100 x 70 pixel. All positive images are placed inside the folder positive_images

Negative Images: 600 Negative images are provided for this training which are collected from git repository of of Joakim Soderberg [Sod14]. All images are resized into 640x480 pixel. These negative images are placed inside the folder negative_images

Step 1: Creating Positive and Negative description files
At first we need to create description file for both positive and negative images, which contain path location for each image per line.

```
find ./negative_images -iname "*.jpg" > negatives.txt

find ./positive_images -iname "*.jpg" > positives.txt
```

Step 2: Creating samples
In this step, we generated samples using opencv_createsamples, which will be used for cascade training. opencv_createsamples can generate large number of positive samples from little number of provided positive images by applying distortions and transformation. So to get more accurate result, Naotoshi Seo wrote a nice perl script including a tutorial [Seo], which help to create better sample from relatively small number of input. This script createsamples.pl create requested number of samples (1500 in our cases) by combining each positive image with random negative images and then run the command opencv_createsamples. The command are given below:

```
perl bin/createsamples.pl positives.txt negatives.txt samples 1500
"opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1
-maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 100 -h 70"
```

we have created 1500 sample which are saved under sample folder, where our positive image size is 100x70. Now we need to create description file for all sample files, which will be readable by opencv_trainingcascade

```
python ./tools/mergevec.py -v samples/ -o samples.vec
```

Step 3: Cascade Classifier Training
We can train our cascade classifier using OpenCV in two different ways: opencv_haartraining and opencv_traincascade. For this thesis, we have used opencv_traincascade, as it is the updated method and allows us to run the process in different thread, which speed up the training.
We have trained our cascade classifier in 20 stages, with 1000 positive samples and 600 negative samples. Due to limitation of memory, we have used 2048 mb size each for

precalcValBufSize and precalcIdxBufSize. We have found that though we have 8GB memory size, and sum of precalcValBufSize and precalcIdxBufSize is 4096 which is half of the total memory sized, but because of some other process, it needs more memory beside this allocated memory. And more allocation than the provided size results computer hanged. The command for the training is given below.

```
opencv_traincascade -data classifier -vec samples.vec
  -bg negatives.txt -numStages 20 -minHitRate 0.999
  -maxFalseAlarmRate 0.5 -numPos 1000  -numNeg 600
  -w 100 -h 70 -mode ALL -precalcValBufSize 2048
  -precalcIdxBufSize 2048
```

During the training, all the xml for each stage (stage 0 to stage 19) are stored in the folder classifier. At the end of the training, it generates a file named cascade.xml, which we have used for object detection.

The object "television" is different from some other object because if television is switched off, the screen become black with its rectangle border. But during the television is on, the screen display different objects depending on the running video. Inside the rectangle border of the television, there can be different colors with different types of other objects such as human, nature, car, mountain etc. Therefore, in our experiment, we have trained television in two ways 1) considering only switched off television as positive image and 2) considering both television off and on as our positive images. The result of the performance in both cases are described in the evaluation section. 6.

5.2.2 Class: WebXRExperiment

This is the main class, which is extended from the XRExamplebase class of webxr-polyfill framework. To implement this class we have taken the help from the example of webxr-polyfill face detection example [Moz18]. In this class we have implemented the main functionalities such as TV / monitor detection, labeling after detection, virtual object creation etc. We have modularized the class based on different tasks. We are explaining here different methods as per their uses.

TV/monitor detection

The first task of this application is to detect television. We have implemented this functionality using opencv library. To detect the television or monitor, the first thing we will need is the cascade classifier of the object, which we have generated using haar cascade training application of opencv 5.2.1

```
this.worker = new Worker ("../../libs/worker-tv.js")
```

Once worker is initialized and loads all the dependency in a different thread, it keeps sending message through an array (sdObjectRects), which contains the information

about detected object. The size of array depends on the number of objects have been detected in that frame. The size of array 0 indicates no object found. Each array element contains the position (x,y) and the width and height of the detected object. Using this value we can draw the rectangle on the screen.

```
cvImageCtx.strokeRect(rect.x, rect.y, rect.width , rect.height);
```

Figure 5.2: Screenshot of TV Detection and control

Once TV/monitor is detected, the create3dObject function is called for further action.

function create3dObject(posX, posY) The purpose of this function is to create virtual remote and 3D text in the desired position. Virtual remote allows user to communicate with the television in terms of volume up/down, channel change, video source changes etc. which is create by the function this.createRemote() and 3D text shows the information provided by application which is created through the function this.createText(). This function create3dObject is called with the parameters x and y, which are position value of the object in the screen frame, so that these 3D objects can be created in front of the user.

```
this.session.hitTest(x, y).then(anchorOffset => {
  if(anchorOffset === null){
```

```
    this.msgScore.innerHTML = 'miss'
  } else {
    if(!is3DRemoteCreated) {
      this.addAnchoredNode(anchorOffset, this.createRemote());
      this.addAnchoredNode(anchorOffset, this.createText());
      is3DRemoteCreated = true;
    }
  }
})
```

function createText())

This function is responsible to display some information about the detected object. It creates 3D text using Three.FontLoader() libraries beside the object. We have worked in details on this feature in the application of Education purpose.

function createRemote())

This function is responsible for creating virtual 3D remote, which is called just once first time the system detect the television. At first we initialized a two-dimensional array of grid x and y. All the remote buttons are placed in a position of the grid. In this app, we have created grid 5 X 2 (row by column), which is dynamically adjustable by the changing the value of noOfRemoteRow and noOfRemoteColumn at declaration. In each grid, a 3D button is created using threeJS library and a two image is used as texture for each button using Three.TextureLoader() function. Each grid hold an object of RemoteGrid(), which hold the value for grid no, 3D position of the grid and the value of row and column.

```
for (var i=0; i<noOfRemoteRow; i++) { //height // row
  for (var j=0; j<noOfRemoteColumn; j++) {
    grid[i][j] = new RemoteGrid()
    grid[i][j].gridNo = i*noOfRemoteColumn+j;
    grid[i][j].gridPositionX=boardPositionX+i*(gemsLength+gap);
    grid[i][j].gridPositionY=hightDown+j*(gemsLength+gap);
    grid[i][j].gridX = i;
    grid[i][j].gridY = j;
  }
}
```

gridNo is used later to determine the function need to be execute on a button press as per the array of method

```
var method = ['power','source','tv','hdmi','volumeUp','channelUp',
      'volumeDown','channelDown','mute','previousChannel']
```

Touch Detection

As we mentioned that the detected object can be either television or monitor. Therefore we have implemented two different feature on touch on the screen. First consider the detected object is television. In this case, users will be able to control remote using 3D virtual Remote. In mobile, only 2D surface is available depending on the mobile resolution. But the virtual remote is created in 3D space. When user touches on the mobile screen, api gives us 2D position (x,y) through addEventListener function, which is different from the position of the 3D object in 3D space position system (x,y,z). Therefore, when user will try to touch on a virtual object, we need to understand which 3D object should response according to the touch location. We have solved this problem by calculating the 2D position of 3D object and matching it with 2D touch location. We have discussed this solution in details in the section 5.5 (Web-based XR Board Game): "Select Object by Touch", as in game we have used the same solution.

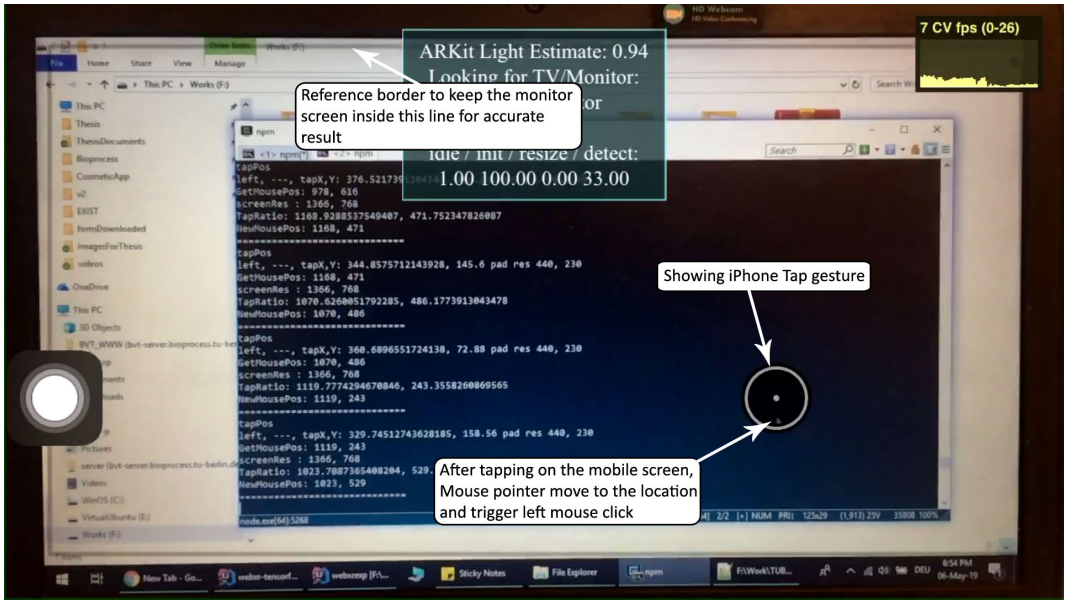

Figure 5.3: Screenshot of laptop mouse control through mobile touch screen

function _onTouchStart(ev)

This function is called when user touch on the mobile screen. The variable ev contains the x,y position of the screen which depends on the mobile screen resolution. This function calls two different function considering two different scenario, i.e TV and computer monitor. To interact with virtual remote, the function *remoteTouch* is called.

```
this.remoteTouch(ev.touches[0].clientX, ev.touches[0].clientY)
```

To use mobile screen as touch screen of the monitor, the function *press* is called

```
const touchWinToDivX = ev.touches[0].clientX *
                cvImageDiv.width / window.innerWidth
const touchWinToDivY = ev.touches[0].clientY *
                cvImageDiv.height / window.innerHeight


// we substract offset to send the position to laptop
var touchInBoxX = touchWinToDivX - mousePadOffsetX;
var touchInBoxY = touchWinToDivY - mousePadOffsetY;

if (touchWinToDivX < mousePadOffsetX ||
    touchWinToDivY < mousePadOffsetY ||
    touchWinToDivX > mousePadOffsetX + mousePadWidth ||
    touchWinToDivY > mousePadOffsetY + mousePadHeight) {
        consoleMsg = consoleMsg + " <> tap outside"
}
else{
    this.press('tapPos', 'left/' + touchInBoxX + '/' +
    touchInBoxY + '/' + mousePadWidth + '/' + mousePadHeight)
}
```

From the above code, we can see that, when we calculate the touch for virtual remote, we send the touch position (x,y) as it is. Because the virtual object is rendering on the mobile screen, therefore the 2D position of the 3D position also will be the same as touch surface.
But to control the mouse position, we need to calculate the touch position in different way, because the resolution of the mobile and the resolution of the monitor is not same. Moreover this, the video canvas resolution and mobile screen resolution is also not the same. After monitor detection, the rectangle has been drawn on the video canvas. Therefore, the touch location inside the video canvas can be calculated through the expression

```
const touchWinToDivX = ev.touches[0].clientX * cvImageDiv.width
                        / window.innerWidth
const touchWinToDivY = ev.touches[0].clientY * cvImageDiv.height
                       / window.innerHeight
```

Now to calculate the touch location respect to the monitor appears in the screen, we subtract the value from x,y position of the rectangle, which is stored in mousePadOffsetX and mousePadoffsetY. At the end, we check that if the touch location is inside the rectangle or not. If the touch is inside the rectactangle, it calls the function press with some parameters such as method tapPos (indicates to call the function to mouse pointer move), left (indicates left button click), touchInBoxX and touchInBoxY (indicates the touch location, where mouse pointer needs to move) and mousePadWidth, mousePadHeight represent the height and width of monitor according to the mobile video canvas.

As resolution of monitor is different than the mousePadWidth and mousePadheight, we can simply calculate the touch position according the resolution of real monitor with this information.

function remoteTouch(locationX,locationY)

Once player touches the mobile screen, this function is called through the function _onTouchStart, where the x,y touch coordinate passes through locationX and locationY. This touch coordinate is compared with 2D coordinate of every 3D object i.e. the location of every button of the virtual remote, which is generated by the function toScreenPosition(). If it matches with any of the 3D object, it calls the function remote3DButtonPress(gridTouch), where gridTouch is the value of grid[i][j].gridNo.

function remote3DButtonPress(gridNo)

This function is responsible for checking the gridNO and validate this number with the array method, if the gridNo is available in integrated method, it calls the press function with the parameter of method. Our implemented method for the television are source, tv, hdmi, volumeUp, channelUp, volumeDown, channelDown, mute and previousChannel.

press(method, param)

This function is used to communicate with TV or computer. In case of television, this function is called from remoteTouch function, with the method parameter only. But in case of computer, this function is called with method parameter along with touch location and width, height of rectangle. To control, Television or computer, we need to run another server, which allow us to communicate with the device through the socket. In this project we have attached two another project name TVServer and remote-control-server which should be running in the local network. The IP address with port number should be replace according to the server configuration.

5.2.3 Class: worker-tv.js

This class runs in different thread using the "worker" method provided by nodejs. This class is responsible for loading the openCV and detecting the object using opencv haar cascade method. the cascade classifier for tv/monitor detection cascade_tv.xml is provided to this class. Once openCV is loaded, it takes the video frame and convert the frame image into gray scale so that it can match the haar feature. This gray scale image is provided for object detection against the cascade classifier xml. If the object is detected, it returns the position of the object in the frame (x, y), the width and height of the object.

Once the main class WebXRExperiment receive the parameters from the worker, it draws the rectangle using cvImageDiv and call the function create3dObject()

5.2.4 Class: RemoteGrid.js

This class is used to keep details of every button of the remote. gridNo represesnts the array number that is used to determine which action should be triggered. gridX and gridY represents the two dimensional array position (i,j) of each grid. girdPositionX and gridPositionY hold the two axis position of the button in the 3D environment, the third axis is constant in our app to display the remote in eye suitable position from user perspective and gems represents the button object which is created through threejs.

5.2.5 TVServer

TVserver is another independent application, which works as a bridge between Television and WebXR application. WebXR application send different request such as volume up/ down, channel up/ down. mute etc to the TV server application, and then TVServer execute the request by communicating with the television. This web application is developed based on third party nodejs framework for Samsung API developed by beejacobs [bee17]. This application should be run on any computer in the same network where Television is connected. The IP address of the television should be configured in the file config.json. Once TVserver runs, the host ip address, should be put in the WebXR app, so that virtual remote from XR app can communicate with television through TVserver application.
Once user tap on any button on the 3D virtual remote from XR app, it calls the function *press* with the different parameters such as volumeUP, volumeDown, mute, channelUp, channelDown etc. XR app send the request to the TVServer. Then TVServer execute the request communicating with the television using the television ip address and Samsung API.

5.2.6 Mouse Control Server

Mouse Control Server is the modified version of Remote-Control-Server. Remote Control server is developed by jeremija [jer18] on the top of RobotJS API 2.7.11. This application allows user to control computer from any web browser on any other computer or mobile in the same network. It supports mouse movements, clicking, scrolling and keyboard input.

In this app we have modified this open source project according to our requirements, so that our webXR app can communicate with computer and mobile screen can be used as touch screen of the monitor. This application should be running on the computer, which we want to control. Once this application will be running, we will receive a host address, which should be configure in our webXR app, so that it can send the request to the computer using this web address.

We mentioned in earlier section that, in case of computer, the function *press* send the request to the host address with methods and param. Method contains "tapPos" and

param contains touch position according to the rectangle in mobile screen. Once index.js from remote control server receives the request with this parameters, it calculates the tap position according the resolution of the monitor from the tap position of the rectangle in the mobile screen.
We can get the resolution of the monitor of computer using robotJS api

```
let srcWidth = robot.getScreenSize().width;
let srcHeight = robot.getScreenSize().height;
```

The tap position can be calculated using the equation

```
let tapAsPerResX = req.params.tapX / req.params.mousePadWidth
                   * srcWidth;
let tapAsPerResY = req.params.tapY / req.params.mousePadHeight
                   * srcHeight;
```

Now we can execute the action on the tapped position by moving the mouse pointer and trigger the left button action.

```
robot.moveMouse(tapAsPerResX, tapAsPerResY);
robot.mouseClick(req.params.button, false)
```

5.3 E-commerce

This is the implementation of our second use case: webXR implementation inside a regular e-commerce website (3.1.2). The landing page is a simple e-commerce store with different sunglasses' pictures and their price. When user tap on the button "try it", it enables XR application for face detection and object visualization.

5.3.1 Product Page

The demo e-commerce site has been developed in index.html file. To develop this page, we have used bootstrap for the designing and vue.js framework for JavaScript functional support. Vue.js is a JavaScript framework for building user interface. As a database for the products, we have used simple json format file named products.json which contain the unique ID of each product, the name of the product and price. Once trial button is tapped by user, it refers XR application with productID as parameter.

5.3.2 XR View

Once the trial button from any product is clicked, xrTrial.html is called with productID. This page is responsible for face detection and put the glass on the eyes. productID determine the image of sunglass that needs to display. This Page appears to the user as a modal view, so that clicking out side of the modal view, user can go back to the product page easily. As OpenCV takes time to be loaded, so product list also appears at the bottom of the camera, so that without goin back to product list, user can switch

to any other sunglass in run-time. Note that, as modal view does not work properly with ARKit enabled browser, WebXR Viewer iOS, therefore, in case of mobile, instead of opening in modal view, the xrTrial page open as a new page

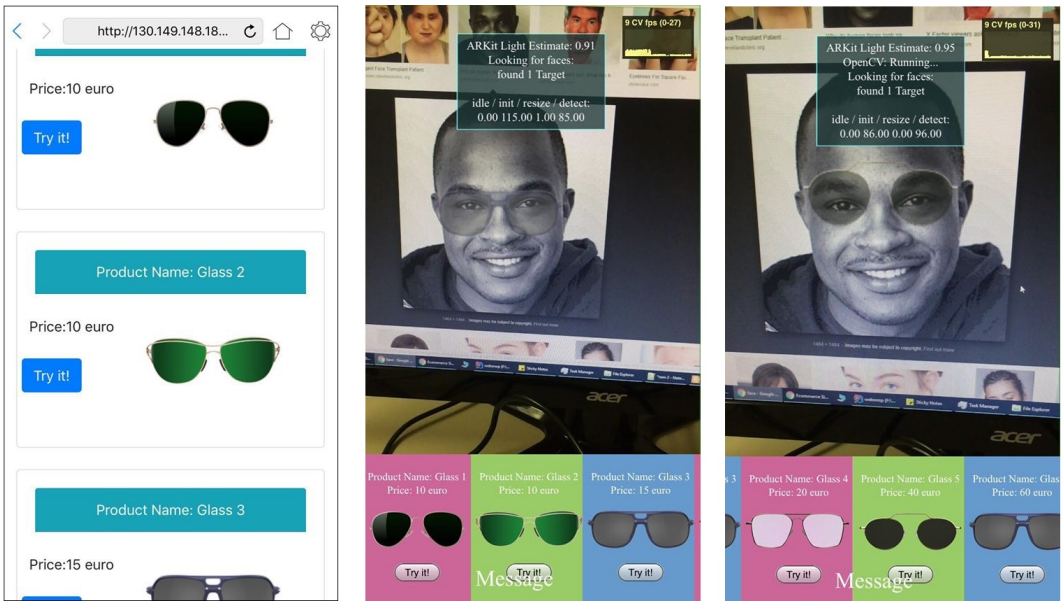

Figure 5.4: E-commerce store and trial

To detect the face, we have used the same class, WebXRExperiment which we have explained in our previous use-case implementation 5.2. To detect Television or monitor, we have created our own cascade classifier, But to detect face, we have used example cascade classifier for face detection provided by openCV, which is also used in webxr polyfil example. [Moz18]. If the system detect the face, we draw the sunglass of selected ID on the eyes.

```
var img = new Image();
img.src = glassImage;
var w = img.width;
var h = img.height;
var sizer = this.scalePreserveAspectRatio(
                w, h, rect.width * 0.9, rect.height);
cvImageCtx.drawImage(img,rect.x+(rect.width/2.0)-(w*sizer/2.0),
                rect.y+(rect.height/3.5),w*sizer,h*sizer);
```

The variable glassImage contains the image path of the sunglass of selected productid

```
glassImage = "images/" + productid + ".png";
```

The function scalePreserveAspectRatio is responsible for the resize the image according to the face size. During resize the image, it maintains the ration of height and width

of the image, so that product image does not get distorted.

```
scalePreserveAspectRatio(imgW, imgH, maxW, maxH){
    return(Math.min((maxW/imgW), (maxH/imgH)));
}
```

The figure 5.4 shows screenshots of this app. The left image shows the sample sunglass store. The next two screenshot shows, how user can give a trial with different sunglasses by selecting any product from the scroll view product list at the bottom.

5.4 Multiple objects detection

This is the implementation of use-case: 3.1.3 where, we have created an application which is capable to detect multiple objects in a video or webcam. Every detected object has been shown in a bounding box with the name of the object. To implement this application, we have used COCO-SSD API[Ten19a], which is developed on the top of the object detection API of tensorflow that use convolutional neural networks.

5.4.1 Class: WebXRExperiment

As same as other application, this is the main class of this application. To detect the object, it requests video frame from webxr-api. But as tensorFlow coco-SSD api accept only HTMLElement or imagedata as an input for analysis, so XRVideoFrame can not be used as input for object detection. Hence, we converted each video frame into the image data. To do this, we converted each frame of the XRVideoframe into Unit8Array and set the data into the a html canvas, which is sent to the xr-tensor-bundle class for analysis. Once xr-tensor-bundle class return the objects with the detected object name and the value of bounding box, we draw the bounding box around the object and wrote the name of the object on this new html canvas. End of this process, we again request for current video frame from webxr-api to detect the object for next frame. With the function _onTouchStart, if it detects the user's tap on the name of the object, it redirects the site to the Wikipedia that contains more information about the object.

5.4.2 Class: Xr-tensor-bundle.js

This class is responsible to detect the object. At starting of the app, it loads all the pre-trained models of coco-ssd. The function xrTensorDetectionConsole receives the image data and after detecting the objects it returns the array of the objects with the detected object name and the value of the bounding box (x, y, width and height). This class is generated from xrTensor.js through webpack. As webxr api runs on client side, therefore, to analysis the video frame captured by the camera, coco-ssd and tensorflow api needs to be loaded into the client side. Therefore we have created a library named xr-tensor-bundle.js using webpack, which combines xrTensor.js with all dependent JavaScript files i.e. tensorflow api, coco-ssd api into one bundle file.

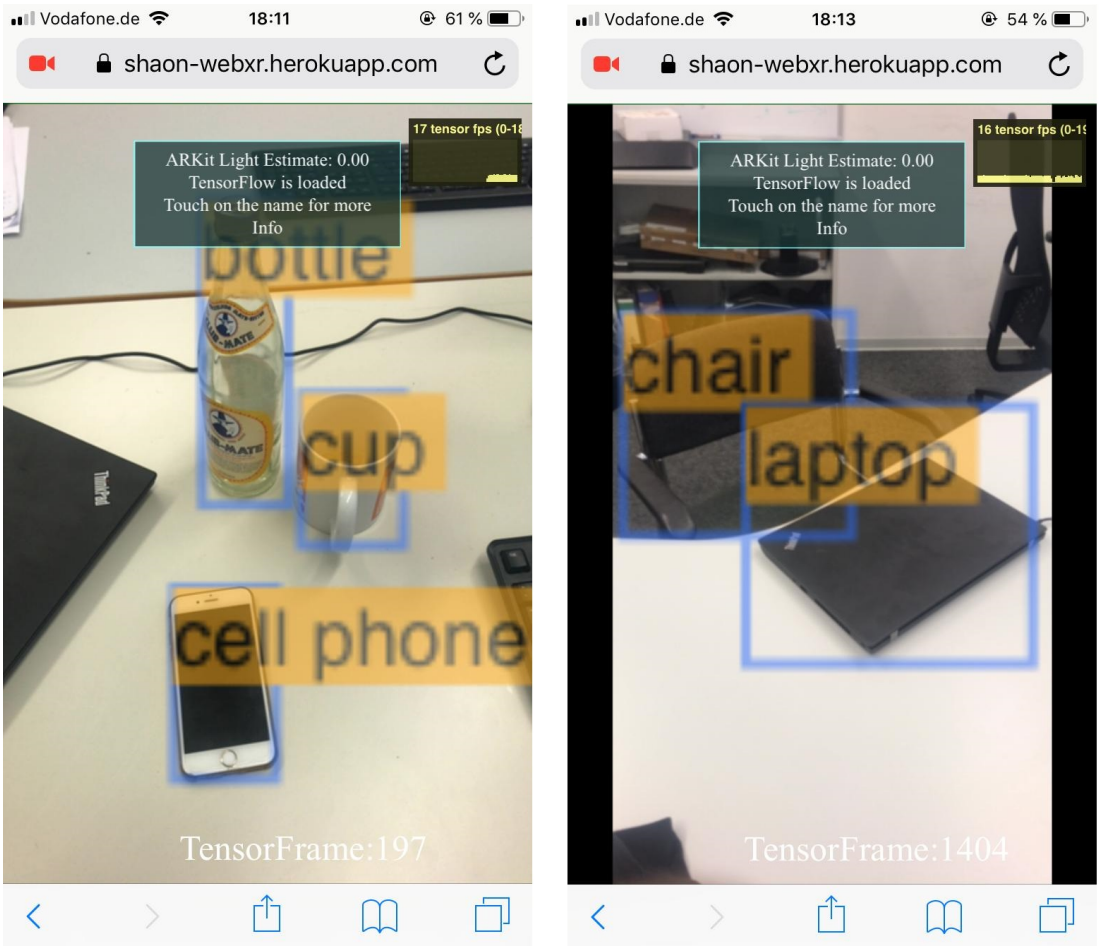

Figure 5.5: Screenshot of multiple object detection using tensorflow

5.5 Web-based XR board game

To develop this XRGame, a main custom class WebXRExperiment was created using the WebXR-pollyfill framework, which interacts with other custom classes for different functionality and also with the third party libraries.

5.5.1 Class: WebXRExperiment

This is the main class of our game, which is extended from the XRExamplebase class of webxr-polyfill framework. Here we have implemented our main game functionality i.e. touch, human turn, computer AI turn etc. We have modularized the class based on different tasks. We are explaining here different methods as per their uses.

Object Creation

To develop our game, at first we need to initialize a scene where virtual world and real world lies together. In the virtual world, we need to create 3D objects such as board, gems etc. The following method is used at the starting of the game.

- **createBoard()** After initializing the scene, this function has been called to create the board at starting. It is called just once. Here we initialized a two-dimensional array of grid x and y. This grid represents the board. In the game we have used 6 x 8 (row by column). But it is dynamically adjustable by changing the value of gridSizeWidth and gridSizeHeight at declaration.

  Here is the code, that creates the board and also initialize the value of array, grid[][].

  Listing 5.1: Code for Virtual board creation

```
createBoard(){
  plate.position.set(0, 0.6, -1);
  this.floorGroup.add(plate)
  for (var i=0; i<gridSizeWidth; i++){//gridSizeWidth =6
    for (var j=0; j<gridSizeHeight; j++){//gridSizeHeight=8
      const box = new THREE.Mesh(
        new THREE.BoxBufferGeometry(gemsLength,
                                    0.05,gemsLength),
        new THREE.MeshPhongMaterial({color: '#ccc0b4'})
      )
      box.position.set(boardPositionX+i*(gemsLength+gap),0,
                       hightDown+j*(gemsLength+gap))
      plate.add(box)
      grid[i][j]=0;
    }
  }
  grid[0][7]=1;
  grid[5][0]=2;
}
```

  From the above code, we can see that while creating the board, grid[i][j] were set as 0, where 0 represent empty block, But later we defined grid[0][7]=1 and grid[5][0]=2, where 1 represent blue gems and 2 represents red gems, So after creating the initial board, we called loadGems(), where according to the value of array grid[][], i.e. 0, 1 or 2, blue or red gems will be placed on the board.

- **loadGems()** Through this function we load the gems and place them in their desired positions. This function is called after each turn of human and computer.

The array, grid[][], keeps track of gems on each of the blocks. Also, this function is responsible for calculating the score after each turn.

Select Object by Touch

In the previous use-case 5.6, we have detected the touch using raycaster feature provided by three.js library. In this application we have detected object from the touch through different way by our own algorithm.

In mobile we have only 2D surface depending on the mobile resolution, such as 750 x 1334 (for iphone 6, 6s, 7 etc.). But in the virtual world, components are positioned in 3D space. When we touch the mobile screen, api give us 2D position thorough default addEventListener function. In our solution we calculate 2D position for every virtual 3D component based on the touch of the player on the screen. To get this 2d position of 3D object we have developed toScreenPosition function. If the touch position and 2d position of a particular 3d object matches, we invoke the subsequent functions for that 3D object accordingly.

- **_onTouchStart()** When user touches the screen of the mobile, this function is called from the default function addEventListener. This function stores the touch coordinate (x,y) in an array _tapEventData.

- **toScreenPosition()** When the player touch on the screen, this function returns the 2D coordinate of virtual 3D object for that particular moment. Note that, this position can change any moment, as 3D component change the position on the mobile screen with the movement of the device.

- **updateScene(frame)** This function is defined by framework, which is called in every frame. In our game FPS (Frame per second) is 60, that means this function is called 60 times in a second. If player touches the screen in one frame, i.e. if _tapEventData is not null, it calls touchBegin function to calculate player's turn.

Player's Game

In this part we discussed about the functionalities which is running when player is giving his/her turn. Once Player see the board with gems in mobile screen and it is player's turn, Player should be able to touch on any blue gem. If player touch on the blue gem, blue gem will be scaled up to make player understand that, the gem is selected. Once a gem is selected, the system will show yellow and blue blocks as a hint for available moves. If player touch any of these blue or yellow blocks, system will do the rest on behalf of the player, i.e. moving gems to the desired block, changing red gems to blue

gems from adjacent blocks if applicable. Player can deselect the gem by tapping on the same gem once again, if s/he decide to select any other gem.

- **touchBegin(locationX, locationY)** This function is the most important function when player gives his/her turn. This function executes if it is not computer's turn. Once player touch the mobile screen, system receives x,y touch coordinate. This x,y coordinate is compared with 2D coordinate of every 3D object, which is generated by the function toScreenPosition(). If it matches with any of the 3D object, it checks if the 3D object is one of the blue gems. If not, no action is taken and waiting for next touch attempt. If the touch attempt is failed more than 2 times, It gives a message "Get closer to the gem for better touch." If any blue gem is selected, it shows the hint by yellow and blue color to the player by calling showAvailableMove() function. Once player touch any hinted block, it shows the animation using tweenjs and it changes the value of the array grid[][] of adjacent block's position from 2 to 1. At the end of tweenjs execution the function playerTurnAction() is called to switch the turn from player to computer.

- **showAvailableMove()** Once player touch a blue gem, this function is called to show the hints (yellow and blue) for available move. When any blue gem is selected, the value of grid[][] for all available allowed blocks are updated. If it is adjacent block, grid[][] value is set as 11 (for blue) and if it is 1 block far from the selected gem, grid[][] value is set as 111(yellow). Then it calls reloadgems() to show the hints i.e yellow and blue color on the board.

- **playerTurnAction()** This function is used to switch over the turn from player to computer. It calls reloadGems() function, to load the updated status of blue vs red gems after player's turn. It also disables touch for players, so that player can not touch the board or gems during computer's turn. At the end it calls computerTurn() function to call Artificial Intelligence implementation.

- **reloadGems()** Once player give his/her turn by selecting any yellow or blue hinted block, this function is called. This function removes all the gems objects from their parents, and call loadGems() function to reload the gems according to the updated array grid[][]. Grid type 1 represents player (blue gems), grid type 2 represents computer (red gems), grid type 0 represents empty block. There are two more temporary grid types 11 and 111. Grid type 11 represents blue hints and grid type 111 represents yellow hints.

Figure 4 shows one of the moment during the game, when player select a gem and get hints for available movement.

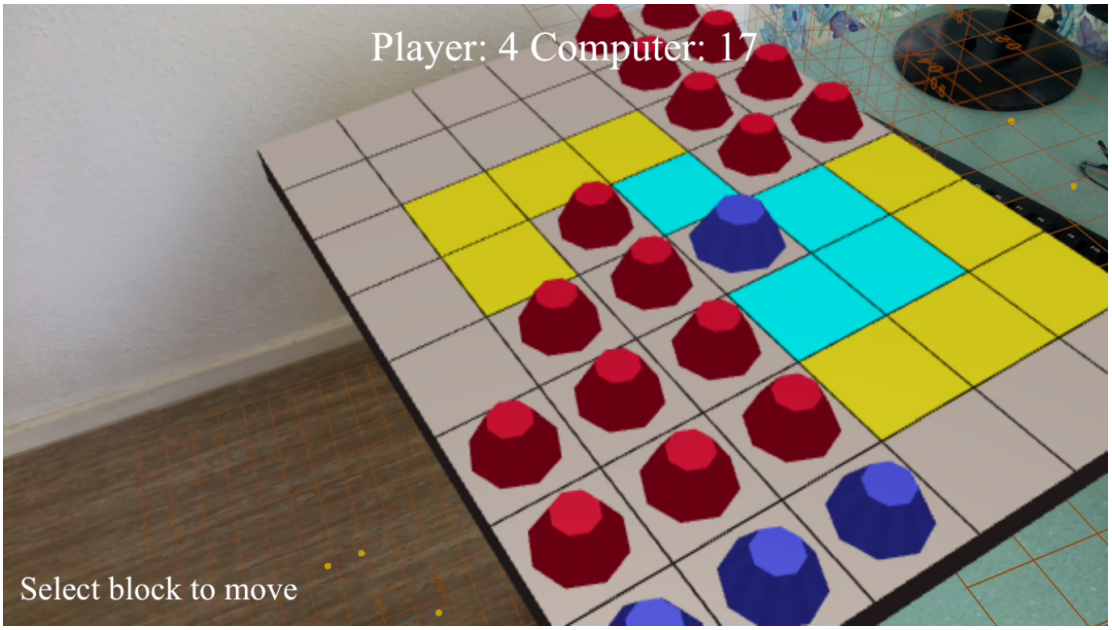

Figure 5.6: Screenshot of the game

AI Implementation

Once player's turn is completed computerTurn() is called from playerTurnAction(). The artificial intelligence has been developed in this section, where computer can give the best possible move against the player.

- **computerTurn()**

  This is the first function which is called after human's turn. In this function we have implemented computer AI, using BestMovement Class. Here we calculate the best movement for every gems of computer and store most beneficial move within first adjacent neighbor blocks and then in same way we store most beneficial move within second adjacent neighbor blocks. Then we find out the most beneficial move between them. The "most beneficial move" is determined by checking the number of blue gems at the adjacent block of the intended block. If the number of blue gems is 0, it simply select the first red gems of the array and select the first available adjacent block as intended block. In both the above mentioned cases, the selected gems's original block position and the intended block's position are saved in BestMovement class. Then it calls for the computerTurnAnimation() function.

- **computerTurnAnimation()** After taking decision of the turn, from computerTrun() function, this function just show the animation of computer's turn i.e. movement of red gems from original position to intended position. Once animation is over, it calls computerGiveTurn().

- **computerGiveTurn()** This function is responsible for updating the array, grid[][] which keeps track of gems position on the grid for both players and then it calls turnExchangePctoPlayer function

- **turnExchangePctoPlayer()** As now grid array has updated value, it calls reloadGems() function to update status of blue and red gems. then it changes the computer's turn to player's turn by allowing human touch.

Game Over

The last part was to calculate the scores for each player and handling the game over feature.

- **checkGameOver()** Considering the performance, we call this function only if the number of empty block is less than 10 or any of the player's score is 0. We have started to check game over if empty block is less than 10 because, in this moment, the situation may come that, though there are some empty blocks in the board but one of the player does not have any gems which can move to first or second adjacent neighbor blocks as those blocks are not empty (occupied by opponent's gems). As any of the player does not have any available move, we declare "game over" and decide winner according to the current score. It calls gameOver function if no movement is available for any of the player or the score of any of the player is 0 or the number of empty block is 0.

- **gameOver()** If checkGameOver function finds that the game is over, this function is called. This function just disable all the touch, and update user interface by showing the result.

5.5.2 Class: EquipEnemyGrid

This class is used to keep record or details of every gems. i.e., its type (human, computer or empty block), its position, and the gems object.

5.5.3 Class: BestMovement

This class is used during best movement calculation for computer AI. It stores original position of a piece/gem, the target block to move for best result and number of opponent gems around the target block, so that it can be compared with other gems. Once best movement is decided, using this attribute (original position and target block), we

easily can show the animation.

5.6 Education through 3D model visualization

In this application, we have created a page, which contain the list of models and by clicking on any model, users are able to study on that model. To develop this application, we need supports for other XR hardware such as accelerometer, gyroscope of the devices, therefore this application only capable to run with XR Devices. For 3D model, we have used some free 3D model from two different websites https://free3d.com/ and https://www.turbosquid.com/

5.6.1 Model page

This page contains the list of models, which are explorable easily. Developers can easily add new model, just updating database (edu_list.json in our thesis) and creating the site for the new model. In this thesis, we have showed two different models for study i.e basic human body and skull.

The figure 5.7 shows two screenshot of this application for the skull. In left image, user touched on the skull and in right image, user touched on the mandible. The information is showing at the right side of the model and at the bottom. Developers can add more information according to the requirements.

5.6.2 Class WebXRExperiment

This is the main class of each education app, where we have implemented different functionalities i.e. 3D virtual model loading, touch, text loading etc. We are explaining here important methods as per their uses.

- **_onTouchStart()** This function is called when user touch on the mobile screen through the default function addEventListener. In the first step, we need to load a 3D model. 3D model could be loaded during the scene initialization, but as we do not know the direction of the user looking at, therefore, we decided to load the 3D model depending on the touch position of the user, so that user can manually create the model on their preferred position. To load the model, the function *create3dObject* is called, if the value of isObjectCreated is false. This variable determines if the 3D model is already being created or not. If the model is already loaded, then user's touch is used to interact with the part of the body to view the details information.

- **create3dObject ()** This function is responsible for creating the 3D object on the position where users would like to create. If users do not like the position,

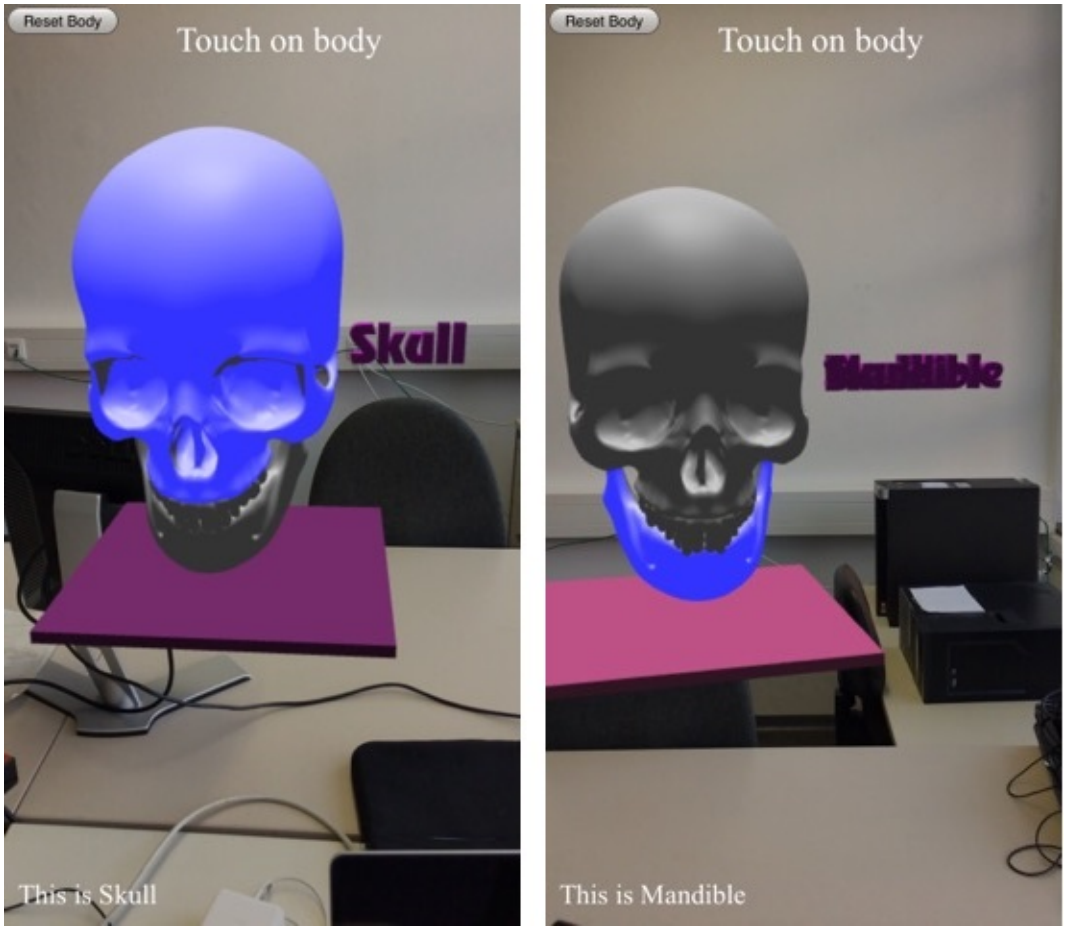

Figure 5.7: Screenshot of the Education App

where 3D object is created, they can remove the model by clicking on the "Reset" button and select the new position. This function create the 3D model with the help of createBody function.

- **createBody()** The 3D virtual model is created through this function. There are different format available for 3D model. In this thesis we have used obj format. The code below shows, how to lead obj file using OBJLoader from ThreeJS library

```
var loader = new THREE.OBJLoader( manager );
loader.load( 'images/skull-basic.obj', function(obj) {
    object = obj;
}, onProgress, onError );
```

In the 3D model, the different part should be drawn separately from each other. Every part should be a different object so that each of them can be detected

separately. Each part that we want to explain or would like to interact by touch, should be an object. In our model for human body, there are seven objects i.e., body, face, hair, right hand, left hand, right leg and left leg). The objects for skull models are mandible, teeth upper, teeth lower and skull. After loading the model using OBJLoader, the model is added to the scene using loadModel() function.

- **touchAction()** This function allows user to interact with the 3D virtual model. In TV-remote touch or in the game, we have calculated the touch on 3D model by calculating 2D position of 3D element and matching the location of tap. But problem with this technique was, the position of a 3D model depend on the position of the anchor point of that model. And user can touch any part of the 3D body, which location does not match with anchor point, though it was still touched on the body. In our game or tv remote, we can calculate height and width of the model easily before matching with user's touch location as it was rectangular. But in this app, models are different in shape. In this application, we have implemented touch detection using raycaster library of Threejs. Once user touch on the screen, raycaster function is called with touch x,y location parameter. During loading the model, we have added all the objects of the 3D model on a place holder name "plate". Therefore, raycaster function check if the touch location intersect with any of the object of the plate:

  var intersects = raycaster.intersectObjects(plate.children);

  If the touch location by the user intersects with any of the object of plate, it gives us the name of the object. Base on the object name we can display respective information of the part of the model by calling the function createText.

- **createText()** This function is responsible to display the information about the selected part of the model. To load the 3D font we have used FontLoader library of ThreeJS.

5.7 Web-based XR game with object detection

This game is also implemented on the top of OpenCV object detection example from webXR-Polyfill framework. In this game to move the bat, we have considered human palm with open finger as controller. The app detects the palm, and move the bat according to the palm position. The figure 5.8 shows one of the moment during the game, when player hit the ball by moving the bat using the palm detection.

To detect palm, at first we need to create cascade classifier for palm using OpenCV haarcascade training that we described in 2.7.5. We have trained the palm and created haar cascade classifier in the same way, that we did for creating classifier for tv/monitor.

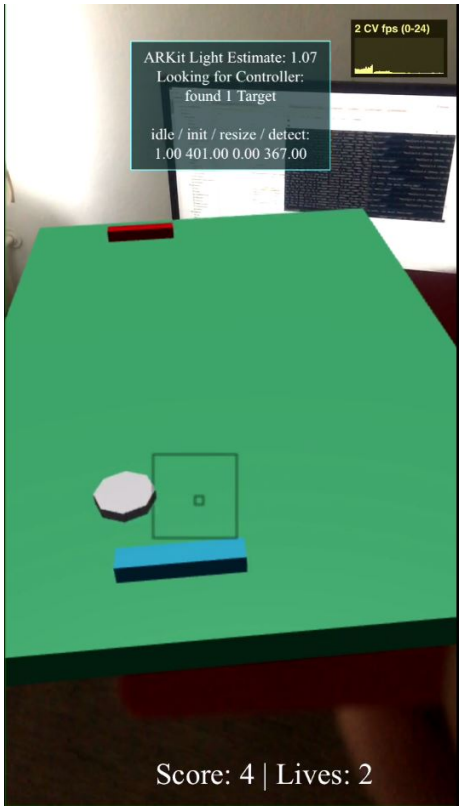

Figure 5.8: Screenshot of webXR Game with Object Detection

5.7.1 Haar Cascade training for palm

We have provided 100 positive images of human's open palm including male and female's left and right hand. All images were resized with 100 x 70 pixel. We have used same 600 Negative images for this training which are collected from git repository of of Joakim Soderberg [Sod14]. All images are resized into 640x480 pixel. For training, we have created 1500 sample and trained up to 20 stages. Besides our own classifier, we have tested our app with haar cascade palm classifier trained by [OAI18] to recheck the performance of our game.

5.7.2 Class: WebXRExperiment

In this main class, we have implemented our main game functionalities i.e. touch, game play, score calculation etc. This class is also responsible for object detection and react game control according to the detected object position. We are explaining here different methods as per their uses.

Object Creation

At first after initializing the scene, we have created the 3D objects such as board, ball, bats in the virtual world. The following method is used at the starting of the game.

createBoard() After initializing the scene, this function has been called to create the board at starting. It is called just once. After the creation of the board, we have create another three virtual objects i.e. two bats in both side for player and computer, and the ball. To move the player's bat through x axis, we have created one dimension grid from left to right of the board. According to our board width, 26 grid is good enough to move the bat to any required position. But this number can be increased using the variable boardwidthCell.

Here is the code, that creates the board and also initialize the value of array, grid[][].

Listing 5.2: Code for Virtual board creation

```
createBoard(){
  plate.position.set(0, 0.6, -1);
  this.floorGroup.add(plate) // Create Board

  basePlayer.position.set(0,0.0,0.5)
  plate.add(basePlayer); // create Bat for player

  baseComputer.position.set(0,0.0,-0.5)
  plate.add(baseComputer); //create bat for computer

  ball.position.set(ballX,0.0,ballY)
  plate.add(ball); //create ball and position it in the middle

  //Create the grid through X-Axis so that, bat can be moved -
  //in any of this position
  for (var i=0; i<boardwidthCell; i++) {
    gemsGrid[i] = new BoardGrid();
    gemsGrid[i].gridPositionX=boardPositionX+i*(gemsLengthX+gap);
    gemsGrid[i].gridPositionY=hightDown+boardPositionY;
    gemsGrid[i].gridX = i;
    gemsGrid[i].gridY = 0;
    gemsGrid[i].type = 0;
    gemsGrid[i].gems = new THREE.Mesh(
      new THREE.BoxBufferGeometry(gemsLengthX,0.0,gemsLengthY),
      new THREE.MeshPhongMaterial({color: '#52c682'})
    )
    gemsGrid[i].gems.position.set(boardPositionX+i*(gemsLengthX
                                  + gap), 0,hightDown+boardPositionY)
    plate.add(gemsGrid[i].gems);
  }
```

```
  isBoardCreated=true;
}
```

Palm detection

After creating the board, now next task of this application is to detect palm, so that game can be played using hand. We have implemented this functionality as same way as we implemented TV/Monitor detection using opencv library and worker. To detect the palm, the first thing we will need is the cascade classifier of the object, which we have generated using haar cascade training application of opencv 5.7.1.

Once worker is initialized and loads all the dependency in a different thread, it keeps sending message through an array (sdObjectRects), which contains the information about detected object. If palm is found, it sends the position (x, y) and the size of palm (height, width). Using this value we have draw the rectangle and the center of this rectangle on the screen so that the player can easily understand where to put his/her hand on a position to move the bat.

```
cvImageCtx.strokeRect(rect.x, rect.y, rect.width, rect.height);
var locationX = rect.x + rect.width/2.0;
var locationY = rect.y + rect.height / 2.0;
cvImageCtx.strokeRect(locationX, locationY, 5 , 5);
```

Move Object by palm detection

After detection of the palm, we get the 2D position of the palm according to the mobile screen resolution. Therefore, we need to calculate that the corresponding 2D position of the board, so that we can move the 3D bat to the intended position. In our solution we calculate 2D position for every grid according to the mobile screen, which were created during creation of the board. During matching the position, we have only considered the position x, as the bat is only allowed to move x axis. If the x location of the palm is smaller than x position of grid[i+1], we have moved the bat position to the position of gird[i] and stop the loop. Here is the code for movement of the bat according to the palm position

```
outerloop: for (var i = 0; i < boardwidthCell-1; i++) {
  // get 2D position of the grid
  var objPos = this.toScreenPosition(gemsGrid[i].gems)
  // locationX is the position X value of detected palm which -
  // is comparing with position x of a grid
  if( locationX <= objPos[0]){
    basePlayer.position.set(gemsGrid[i].gems.position.x,
                            gemsGrid[i].gems.position.y,
                            gemsGrid[i].gems.position.z)
```

```
    break outerloop;
  }
}
```

Game Play

Ball Move: The ball changes it's position in every frame of the game. Initially in every frame we change the ball position as ballX += dx and ballY += dy to move the ball, where dx and dy is initially set as 0.01 and -0.01. To make it harder, after every 10 points, we have increased this dx and dy value. If the ball touch the edge of the board in -x or +x direction, we have changed the dx value to -dx to change the direction of the ball. In the same way, if the ball touch the bat of player's or computer which are placed in +y and -y direction of the board, we changed the value of dy to -dy to change the direction of ball movement.

Score and lives: The player should touch the ball using his bat, before the ball touches the edge of board in +y direction. If player successfully hit the ball, player score will be increased by one but if he/she fails to hit the ball, which means, when ball reaches to the edge of the board, but it founds that, the ball's x position is outside of the bat's width, one live will be decreased.

```
if(ballX >= basePlayer.position.x-(0.1+ballRadius) &&
   ballX <= basePlayer.position.x+(0.1+ballRadius)){
        dy=-dy;
        score++;
        if (score == gameWinScore)
                this.gameOver(1)
}
else{
    lives--;
    if (lives == 0)
            this.gameOver(0)
        else
                this.resetBall()
}
```

To win the game, player needs to make 30 score. If player loose 3 lives before reaching this score, he/she will lose the game. Every time player loses the live, it call resetball function. In the function resetBall(), it sets the ball position in the middle of the board and direction of y axis is set to negative, so that at starting the ball moves to opposition of the player's side. If player score 30 or loose all his/lives, it calls the function gameOver() with parameter 1 or 0, where 1 indicate winning and 0 indicate loosing the game. The function gameOver is responsible for showing the relative message and score according the result of the game.

5.7.3 Class: BoardGrid

This class is used to store the information i.e. position of every grid, which is later used to compare with the position of detected object (palm) and set the position of the bat.

6 Evaluation

As part of the evaluation we performed various kinds of testing during the entire development life-cycle of the each application. Some testing methods are applicable for all application. But some of them are application specific testing as example OpenCV library does not used is all of the applications. The testing methods consist of manual testing. It refers to the process of manually testing the software for defects and bugs. There are several stages of manual testing we performed over the period.

As part of **Unit Testing** we tested an individual unit or group of related units. It was often done by the programmer to check whether the units they have developed and implemented are behaving as per the expectation or not.

In **Integration Testing** we tested a group of components to produce the desired result. We have done this integration testing to check whether different classes are interacting properly with each other or not. How the AI is responding with the game rule we checked it here.

As part of **System Testing** we tested the application in different devices to determine how it works in different environment. As a parameter we checked the latency in both Android and iOS devices.

To check the performance of our application we decided some factors keeping in mind, some metrics like response, environment, faced challenges etc. Our application is mostly for mobile and tablet, so we thought to do the performance testing on different iOS and android devices. Here are some factors on which we have done our evaluation.

6.1 Latency Analysis

For latency analysis we captured the time when the object was touched as the input time and the time, when the object based on the touch responded, as the response time. We calculated the difference between these two timestamps to check the latency. The main reason for the time delay is that, the virtual board and virtual remote are made up of grids and when we touch on the board it will internally check on which grid you have touched thus results into a delay. In Fig 6.1 (left table) we can see the latency readings we have captured by doing this test in iOS device (iPhone7). In Fig 6.1. (right table) the latency readings for Android device (Samsung S9) have been shown. From both the

tables we can see that the average latency in Android device i.e. 4.66 ms is lesser than that of iOS device (7.55 ms). The reason behind this difference can be because of the version of the devices. But from our testing we can see that our app is running faster in Android devices.

iOS		
Input	Response	Latency
1531054039163	1531054039171	8
1531054672806	1531054672813	9
1531054762334	1531054782344	10
1531054868853	1531054868862	9
1531054936697	1531054936701	4
1531055004513	1531055004526	13
1531055062144	1531055062145	1
1531055137301	1531055137305	4
1531055197787	1531055197797	10
Average		7.55 ms

Android		
Input	Response	Latency
1531399859308	1531399859312	4
1531399910788	1531399910792	4
1531399974883	1531399974889	6
1531400000285	1531400000290	5
1531400024166	1531400024172	6
1531400044558	1531400044562	4
1531400075627	1531400075630	3
1531400100713	1531400100718	5
1531400119541	1531400119546	5
Average		4.66 ms

Figure 6.1: Latency analysis between touch and response of Gems (iPhone 7 and Samsung s9)

6.2 Surface Detection

While doing the development we faced issues many times because of bad surface detection. So, depending on different scenarios we evaluated our app based on surface detection. We have performed this testing in both full light and low light condition. After doing the evaluation we found that the accuracy of surface detection is better in full light compared to low light (shown in Fig 6.2). So, while loading the game we should keep in mind the availability of light as well.

6.3 Object Detection

This testing performed only for the application for first use case 3.1.1 to measured the accuracy of television detection. We mentioned earlier 5.2 that, we have trained cascade classifier two times with different condition 1) considering only the image with TV/monitor off as sample data. 2) considering both TV/monitor on and off. The figure 6.3 shows the result for both cases. We have tried to recognize TV/monitor 20 times from different angle with switched off TV/monitor, another 20 times with switched on TV/monitor. We have also considered other objects, which look like television such as computer casing, black boxes etc, From the fig 6.3(a), we can see that it can recognise switched off television with 100% accuracy. Though the cascade classifier is created only

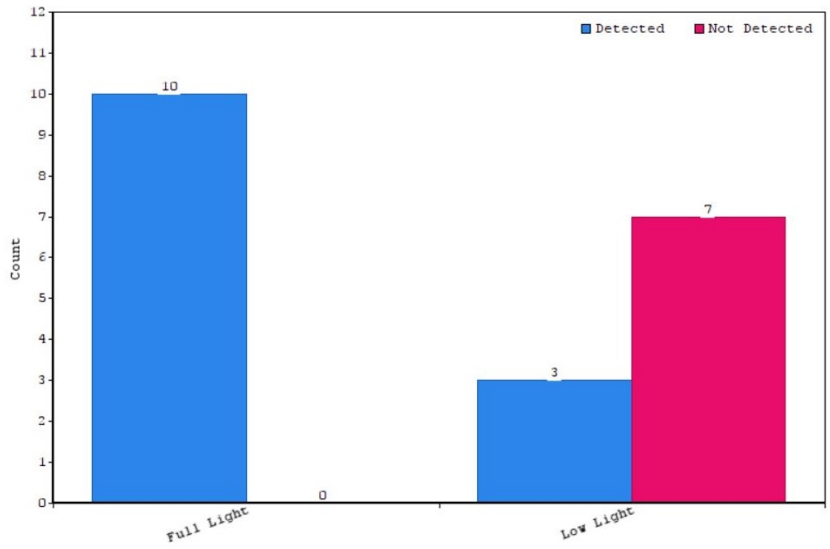

Figure 6.2: Surface detection by Webxr-api in different light condition

with switched off TV/monitor, still it is able to recognise it depending on the scene. Mostly it can recognise, if the on going scene is respectively darker. But we can see that it also recognised four other objects as TV/monitor though they are not. The figure 6.3(b) shows the result with cascade classifier with both TV on and off training sample image. With this classifier, the switched off tv/monitor is identified correctly. But if the television is running, because so different running scene on TV screen, the performance get poor. In the same time, the wrong recognition rate also increased.

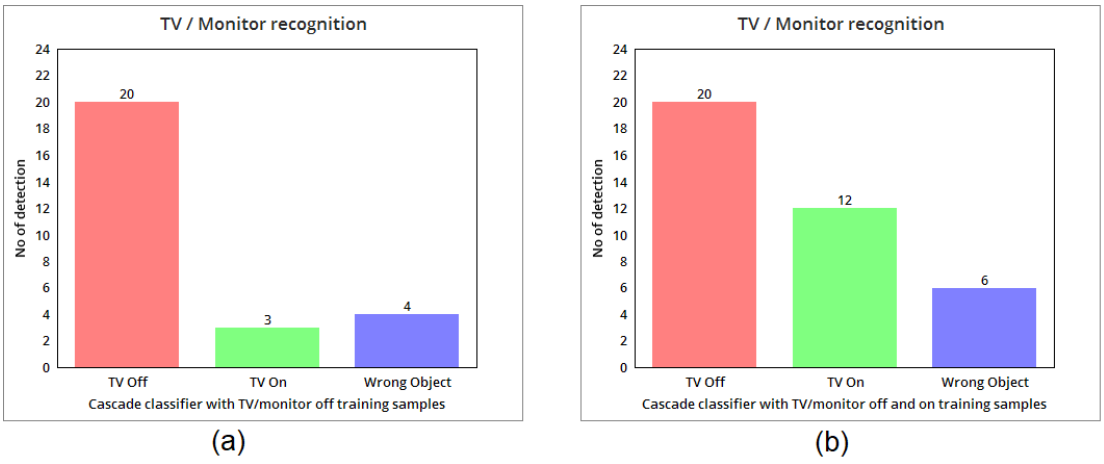

Figure 6.3: TV detection (a) Cascade classifier with TV/monitor off training samples (b) Cascade classifier with both TV/monitor off and on training samples

6.4 Performance of Object Detection

We have implemented object detection using two different ways, 1) using OpenCV: haar cascade classifier, 2) using TensorFlow:COCOSSD. We have measured performance for both of them. We request for a video frame from webXR-API, and send the frame to object detection library for analysis. Once they return the value, we asked for the next video frame. So it may happen that, if object detection is slow, before they return the value, some of the frames have been passed. The FPS for our app is 60. To measure the performance, we have check the FPS of the object detection library.

6.4.1 Using OpenCV: Haar cascade

We have checked the performance in iPhone 6s, iPhone 7 and laptop. In the figure 6.4 (a), we can see the FPS on different devices for openCV. For iphone 6s though the fps is around 7 to 10, but it is well enough to use the application on this device. iPhone 7 came with stronger hardware configuration, there fore, we can see that the the fps dramatically improved in this device. As computer is powerful than mobile, so the fps in laptop is very high. the configuration of the laptop which is used for this performance test is: 2.20 GHz i5 Intel core CPU with Intel HD graphics 5500.

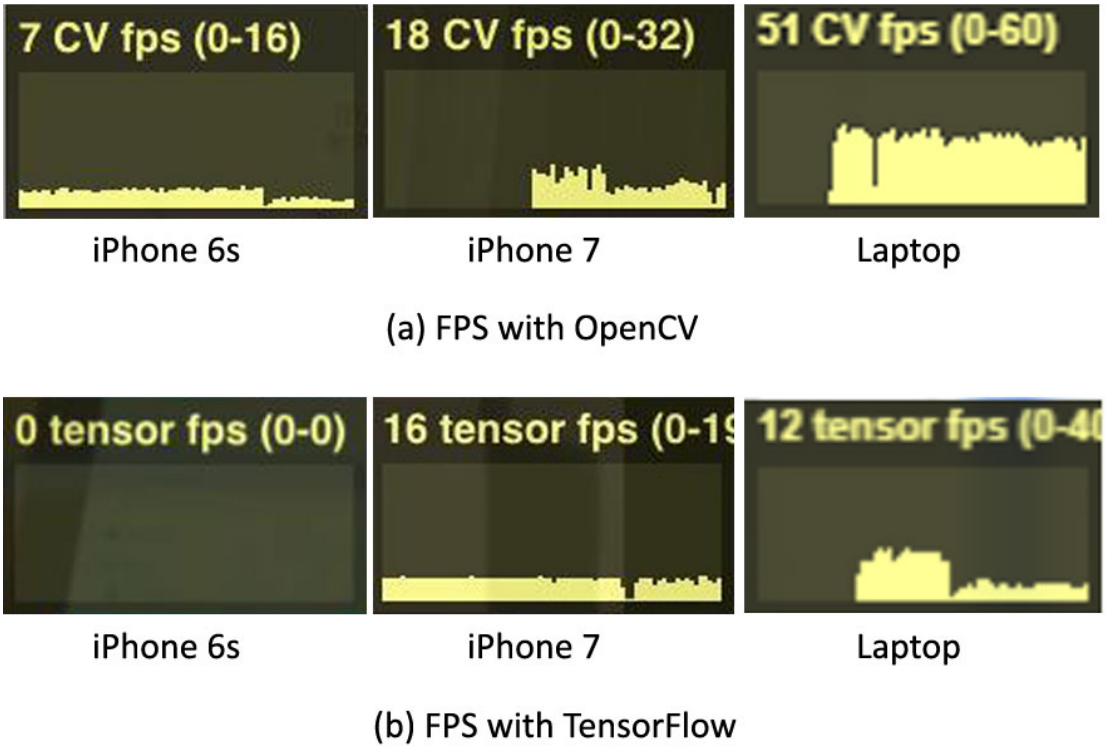

Figure 6.4: FPS with OpenCV vs Tensorflow in for app 3.1.1 and 3.1.3 on different device

6.4.2 Using Tensorflow: COCO-SSD

Tensorflow COCO-SSD takes relatively high processing unit compare to OpenCV. The figure 6.4(b) shows the FPS of the app with TensorFlow. Though the FPS of the application with OpenCV:haar cascade was poor in iPhone 6s, it was still running enough fast to use it as an app. But, the FPS of the application implemented with tensorflow: COCO-SSD was so poor in iPhone 6s that the app got stuck in every frame. This app run well in iPhone 7 or later. The FPS in iPhone 7 was around 16, which is compareable with OpenCV. But the performance in laptop for TensorFlow relatively very poor compare to OpenCV, though it is fast enough for object detection.

6.5 OpenCV performance with Game

We have developed a very basic webXR game with object detection (use-case3.1.6) to evaluate the performance. In general XR app with object detection functionality, we have found that fps was good enough to run the application flawlessly 6.4. And also in other app, if Object Detection calculation miss some frame, it does not effect so much in user experience. But in the game, if object detection performs slowly it may interrupt the game play. Therefore, object detection should be fast enough so that player can control the game elemental whenever it is required. From the figure 6.4, we can see that apps with Object detection using OpenCV, the fps was more than 7 fps in any device. But from the figure 5.8, we can see that FPS is only 2. In the game fps becomes so poor because, there are many other 3D virtual element those needs to be render, hence the game required more processing power than a normal application. And if fps is slow, it may happened that, in a particular frame, when game element needs to be moved, in that frame, OpenCV is still calculating the result of older frame. As a result, player will not be able to play the game properly. In other way, if we update gameplay frame according to the openCV frame, the game will be too slow to play. So to control game element using hand, finger or any other controller detection, the devices need to be more powerful than the current devices we have.

6.6 Touch Accuracy

We have implemented touch on virtual object in two ways, 1) manually by calculating 2D position on mobile screen of the 3D object, and 2) using the raycaster library provided by ThreeJS library. We faced some technical challenges while implementing the touch on the 3D objects manually, but we solved it later on. So, we checked that also as one of the metrics for the performance evaluation. We have done this testing based on four factors. First, we checked by placing our camera just on top of the board and measure how accurate the touches are sensed both when the board is far and close inside the device screen. We can see from Fig 6.5 (a) that touch detection is more accurate, when the board is closer from top. For the second part we placed our camera at approx 45°angle from the board and again tested it when the device closer and far from the

virtual board. We found that in this case also touch accuracy performed better when the device is closer (shown in Fig 6.5 (b)). But in this case the touch miss is more than successful hit when the device is far in this angle.

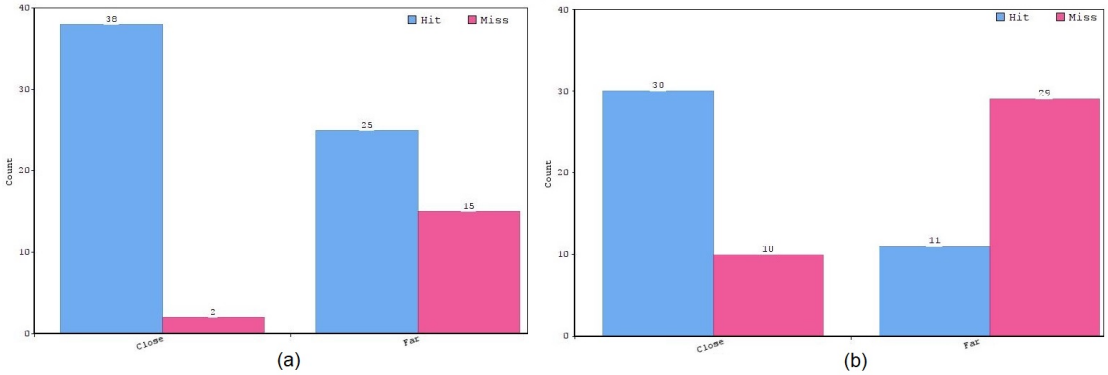

Figure 6.5: Touch accuracy (a) from top and (b) from approx. 45°angle

After this accuracy check we can find from both the figures that touch functionality works better when the device camera is placed on top of the board and when your device camera is closer to the board. It will give you the more accurate results in comparison to other positions and will give you more comfort during playing the game.

6.7 AI Performance

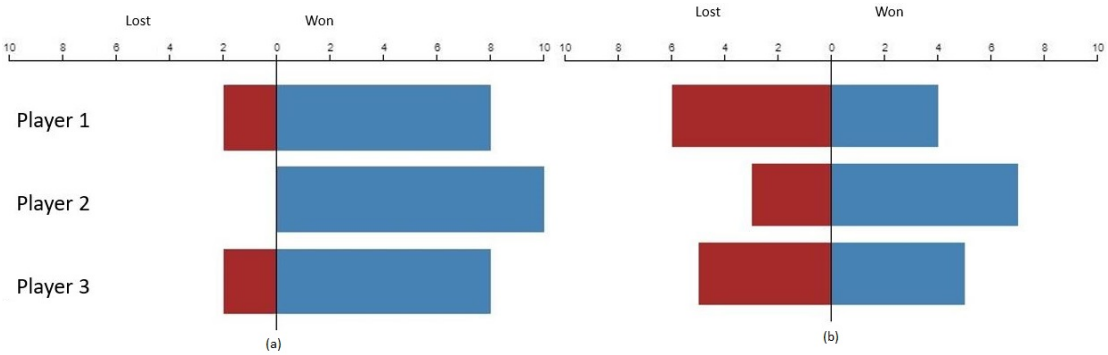

Figure 6.6: Computer's AI performance against (a) Beginner (1st 10 games), (b) Intermediate (2nd 10 games)

This performance testing is only applicable for the use-case 3.1.4. To check how the implemented AI is performing depending on the player's turn we gave the game to three players to play. I would like to thank my friends, who helped to test this game. Each

player played 20 times. We found that when players started playing the game it was really tough to beat the computer, mostly computer won against players (shown in fig 6.6(a)). But once players started to understand the game and gained more experience with the game play, players also managed to win the game against computer. (shown in fig 6.6 (b))

7 Conclusion

This paper gives an overview of Augmented and Virtual reality through the web browser. Though native applications can provide XR experience to the users, but for some use-cases, XR apps through native app distribution model are not appropriate. We have discussed the problems with native XR app, and how these problems can be solved through web distribution of an XR app. But to develop XR apps for web, application needs to access hardware information including sensors, head-mounted display. Also security is another concern for the users. WebXR-device-API allows developers to access all required information through the web browser. This paper describes the generic frameworks and APIs required to develop a web-based XR application including webXR-device-API. We have shown, how web-based XR applications can be developed using webXR-device-API for different use-case including object detection, education, e-commerce and gaming industry. As Object detection is an important part of X reality arena, therefore we have discussed about different object detection technologies and frameworks and how those frameworks can be implemented with webXR-device-API to develop XR application. We have presented the evaluation of our applications on based on different evaluation criteria. During developing and testing phases of our applications, we have encountered different type of issues and challenges, which we have solved in different approaches.

- WebXR-device-API is still an unstable framework, so sometimes it takes time to draw the surface and in some cases, it fails completely. The 3D objects can only be placed properly in the virtual world if the surface can be detected. Therefore, in the app "Education" the model may not be placed properly on touch action sometime as the system still did not find the anchor points. In the same way, in board game, the board can be misplaced sometimes.

  But from evaluation, we have seen that with sufficient light webXR-device-API performs better in terms of 3D space mapping, and the app also works well, if the 3D space is mapped properly. So to avoid unstable issue, users should use the app in sufficient light, and at starting of the session, users should move the device camera slowly around in such a way so that app can map the 3D space in proper way.

- For haar cascade training, the more sample images will be provided, the better classifier we will get. It also depends on the number of training stages. But the classifier training requires huge computational power. For training with our 2 GHz processor with 8 GB memory, each stage took from 2 days upto 2 weeks

(depending on the stage number, in initial stages (1-5), training takes respectively less time than the stages at 20-30). Our cascade classifier for television/monitor has been created by training with 40 stages only, which can be improved by more stages and providing more sample images. To train the model faster, computer with high processing power is recommended.

- In mobile, the app takes a lot of time to initialize openCV. This loading time also depends on the performance of the browsers. In "Safari" browser loading time of openCV or TensorFlow is much faster than the "WebXR Viewer" browser. Because of this loading time, it was not feasible to load XR app again and again for different products in e-commerce website. To solve this problem, we have loaded all the products at the bottom of the XR screen, so that user can easily choose another product to trail without closing the XR session.

- In board game and TV remote, we have implemented touch detection system by converting 3D position of virtual object into 2D surface position according to the mobile screen and matching 2D touch position of the user considering width and height of the object. but still it has some accuracy problem, which we have discussed in evaluation section.

- Rendering during tween animation was also a challenge for us. In case you want to add or delete an object, polyfill framework takes care of that. But in our game, we have to move an object in each frame without removing and creating the object to show the animation for movement. To make that work we have to override the rendering method available within the framework codebase.

- WebXR applications works in ARKit compatible iOS devices (later iOS 11) and ARCore supported Android devices. We were able to do the evaluation in iOS devices but in order to test it in Android devices such as Samsung S9, we had to install Tango Core, which was just crashing while we were trying to launch ARCore supported browser "WebARonARCore".

Our applications are supported in mobile (both in iOS and android) and tablet. The mobile or table must have to be supported by ARkit (iOS) or ARCore (Android). Based on the device type, the supported browser needs to be installed to load the application. New devices are releasing with more powerful processing unit, which will improve real-time object detection. As from evaluation we can see that, though in iPhone 6s TensorFlow performance was so poor to use, but just in next version from iPhone 7 the same application works very smoothly. In the other hand, cloud computing is also becoming new trends nowadays and playing a key role for the applications which need high processing unit. Using cloud computing, heavy computational tasks can be done in the cloud service and API can be used to get hardware information and to render the scene according to the output from cloud service. In this way, XR app can be distributed through web on the devices with low processing power. It also will

save energy and will extend battery life of the device. Though webXR-device-API is unstable but research work and improvement of this API is going on very faster. As the API becomes more stable there will be more scope to make enhancements on these applications.

List of Acronyms

API	Application Programming Interface
AR	Augmented Reality
CSS	Cascading Stylesheets
DHTML	Dynamic HTML
DOM	Document Object Model
FOKUS	Fraunhofer Institut fuer offene Kommunikationssysteme
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JSR	Java Specification Request
QoS	Quality of Service
SDK	Software Developer Kit
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
VR	Virtual Reality Reality
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XCAP	XML Configuration Access Protocol
XDMS	XML Document Management Server
XML	Extensible Markup Language
XR	Represent all kinds of reality including AR, VR, Mixed reality

Bibliography

[App18] Apple: *ARKit*, 2018. https://developer.apple.com/documentation/arkit.

[Bal13] Ball, Thorsten: *TRAIN YOUR OWN OPENCV HAAR CLASSIFIER*, 2013. https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html.

[bee17] beejjacobs: *Node-Samsung-Remote*, 2017. https://www.npmjs.com/package/node-samsung-remote.

[Ber] Berger, Will: *DEEP LEARNING HAAR CASCADE EXPLAINED*. http://www.willberger.org/cascade-haar-explained/.

[CA18] Carpenter, Josh and Reza Ali: *AR: Augmented reality on the web, for everyone*, 2018. https://www.blog.google/products/google-ar-vr/augmented-reality-web-everyone/.

[Fac19] Facebook: *Games on Facebook*, 2019. https://developers.facebook.com/docs/games/gamesonfacebook.

[Goo18] Google: *ARCore*, 2018. https://developers.google.com/ar/discover/.

[Goo19] Google: *Welcome to Stadia*, 2019. https://stadia.dev/blog/welcome-to-stadia.

[Imm18a] Immersive-web: *WebXR Device API Explainer*, 2018. https://github.com/immersive-web/webxr/blob/master/explainer.md.

[Imm18b] Immersive-web: *WebXR: WebXR Device API Specification*, 2018. https://immersive-web.github.io/webxr/.

[jer18] jeremija: *RemoteControlServer*, 2018. https://github.com/jeremija/remote-control-server.

[Kar12] Karpathy, Andrej: *CS231n Convolutional Neural Networks for Visual Recognition*, 2012.

[khr14] khronos: *WebGL and OpenGL Differences*, 2014. https://www.khronos.org/webgl/wiki/WebGL_and_OpenGL_Differences.

[MAC18] MACINTYRE, BLAIR: *WebXR: Experimenting with Computer Vision in WebXR*, 2018. https://blog.mozvr.com/experimenting-with-computer-vision-in-webxr/.

[MMMK03] MATSUGU, MASAKAZU, KATSUHIKO MORI, YUSUKE MITARI and YUJI KANEDA: *Subject independent facial expression recognition with robust face detection using a convolutional neural network*, 2003.

[Moz18] MOZILLA: *WebXR: WebXR-polyfill*, 2018. https://github.com/mozilla/webxr-polyfil.

[Mur16] MURPHY, JOHN: *An Overview of Convolutional Neural Network Architectures for Deep Learning*, 2016.

[Ng] NG, ANDREW: *Convolutional Neural Networks*. https://www.deeplearning.ai.

[Nod18a] NODEJS: *NodeJS: node*, 2018. https://github.com/nodejs/node.

[Nod18b] NODEJS: *NodeJS: Node.js Introduction*, 2018. https://www.w3schools.com/nodejs/nodejs_intro.asp.

[OAI18] OAID: *CVGesture*, 2018. https://github.com/OAID/CVGesture.

[Ope18a] OPENCV: *Cascade Classifier Training*, 2018. https://docs.opencv.org/3.4.1/dc/d88/tutorial_traincascade.html.

[Ope18b] OPENCV: *Face Detection using Haar Cascades*, 2018. https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html.

[Ope18c] OPENCV: *OpenCV*, 2018. https://opencv.org.

[Pac18] PACKET39: *3DOF, 6DOF, ROOMSCALE VR, 360 VIDEO AND EVERYTHING IN BETWEEN*, 2018. https://packet39.com/blog/2018/02/25/3dof-6dof-roomscale-vr-360-video-and-everything-in-between/.

[PAR18] PARACUELLOS, ARTURO: *"WebXR: Progressive WebXR*, 2018. https://blog.mozvr.com/progressive-webxr-ar-store/.

[Pon18] PONNUSAMY, ARUN: *Yolo Object Detection with OpenCV and Python*, 2018. https://www.arunponnusamy.com/yolo-object-detection-opencv-python.html.

[Rea18] REALITY TECHNOLOGIES: *Augmented Reality*, 2018. https://www.realitytechnologies.com/augmented-reality/.

[Rob18] ROBOTJS: *RobotJS*, 2018. http://robotjs.io/.

[Sam18] Samsung: *SamsungAPI*, 2018. https://developer.samsung.com/tv/develop/extension-libraries/smart-view-sdk/api-references.

[Seo] Seo, Naotoshi: *OpenCV Haartraining*. http://note.sonots.com/SciSoftware/haartraining.html.

[Sin12] Singh, Gaganpreet: *Training Haar-cascade*, 2012. https://singhgaganpreet.wordpress.com/2012/10/14/training-haar-cascade/.

[Sod14] Soderberg, Joakim: *haarcascade-negatives*, 2014. https://github.com/JoakimSoderberg/haarcascade-negatives.git.

[Spe18] Speicher, Maximilian, et al.: *XD-AR: Challenges and Opportunities in Cross-Device Augmented Reality Application Development." Proceedings of the ACM on Human-Computer Interaction*, 2018.

[Sum11] Summary of ship movement: *MECHANICAL STRESSES IN MARITIME TRANSPORT*, 2011. http://www.containerhandbuch.de/chb_e/stra/index.html?/chb_e/stra/stra_02_03_03.html.

[Ten19a] TensorFlow: *COCO-SSD*, 2019. https://github.com/tensorflow/tfjs-models/tree/master/coco-ssd.

[Ten19b] TensorFlow: *TensorFlow*, 2019. https://github.com/tensorflow/tensorflow.

[Thr18a] ThreeJS: *Threejs: 3D library*, 2018. https://threejs.org.

[Thr18b] ThreeJS: *Tweenjs: Animation in Three.js using Tween.js with examples*, 2018. https://medium.com/@lachlantweedie/animation-in-three-js-using-tween-js-with-examples-c598a19b1263.

[Thr18c] ThreeJS: *Tweenjs: tweenjs for smooth animation*, 2018. http://learningthreejs.com/blog/2011/08/17/tweenjs-for-smooth-animation/.

[Tow18] Towards Data Science: *Boosting algorithm: AdaBoost*, 2018. https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c.

[Vir17] Virtual Reality Socity: *Virtual reality*, 2017. https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html.

[VJ01] Viola, Paul and Michael Jones: *Rapid object detection using a boosted cascade of simple features*. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[Web19] WEBPACK: *Webpack*, 2019. `https://webpack.js.org/`.

[Wik18a] WIKIPEDIA: *Sense*, 2018. `https://en.wikipedia.org/wiki/Sense`.

[Wik18b] WIKIPEDIA: *Six Degrees of Freedom*, 2018. `https://en.wikipedia.org/wiki/Six_degrees_of_freedom`.

[Wil17] WILLIAMS, OWEN: *AR: WebXR is going to bring VR and AR to the masses. Here's why."*, 2017.

[Woo07] WOODFORD, CHRIS: *Virtual reality*, 2007. `https://www.explainthatstuff.com/virtualreality.html`.

[YFL16] YU, JUNWEI, LU FANG and CHUANZHENG LU: *Key technology and application research on mobile augmented reality*, 2016. `https://ieeexplore.ieee.org/document/7883129`.