A Window to your Smartphone: Exploring Interaction and Communication in Immersive VR with Augmented Virtuality

by

© Amit Prabhakar Desai

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

Sept 2017

St. John's Newfoundland

Abstract

A major drawback of most Head Mounted Displays (HMDs) used in immersive Virtual Reality (VR) is the visual and social isolation of users from their real-world surroundings while wearing these headsets. This partial isolation of users from the real-world might hinder social interactions with friends and family. To address this issue, we present a new method to allow people wearing VR HMDs to use their smartphones without removing their HMDs. To do this, we augment the scene inside the VR HMD with a view of the user's device so that the user can interact with the device without removing the headset. The idea involves the use of additional cameras, such as the Leap Motion device or a high-resolution RGB camera to capture the user's real-world surrounding and augment the virtual world with the content displayed on the smartphone screen. This setup allows VR users to have a window to their smartphone from within the virtual world and afford much of the functionality provided by their smartphones, with the potential to reduce undesirable visual and social isolation users may experience when using immersive VR HMDs.

This work has been successfully submitted for presentation as a poster in the Computer and Robot Vision Conference 2017 in Edmonton, Alberta, and is scheduled to appear in the conference proceedings at the IEEE Xplore digital library later this year.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Dr. Oscar Meruvia-Pastor and Dr. Lourdes Peña-Castillo, for their constant support, guidance, motivation, the dedication of time and knowledge and also their patience during my studies here at Memorial University. This thesis would not have been possible without their encouragement and supervision.

Second, I would like to thank my friend and roommate Pranav Birajdar for his support and generous time in assisting me, by providing test data for my evaluations, whenever I needed them. Also, my gratitude extended to Computer Science General office including Ms. Darlene Oliver, Ms. Sharon Deir, and Ms. Erin Manning, for their support during my master's study. I would also like to take this opportunity to express my deep happiness for having great friends at MUN, without whom this past two years would not have been as adventurous and enjoyable as it was.

Last but not least, I would like to thank my family, especially my parents Prabhakar and Usha and my younger brother Sumit who never stopped loving and supporting me. I owe my achievements to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Virtual Reality (VR) Head Mounted Displays (HMDs) have become increasingly popular among a wider segment of the population [1] thanks to the availability of low-cost hardware and sophisticated game engine software. A major part of the credit goes to products such as the Oculus Rift [2], HTC Vive [3], Samsung Gear VR [4], PlayStation VR [5], Google Cardboard [6] and similar technologies, which have extended the accessibility of immersive VR to a wider segment of the population. These consumer devices provide a rich, interactive and immersive VR environment by offering low-cost, high-resolution stereoscopic images, wide field of view and, in some cases, high graphics quality. In addition, the use of Virtual Reality Head Mounted Displays (VR HMDs) is increasingly being applied in gaming, immersive and collaborative data visualization [7, 8], archaeology [9, 10], training [11, 12], and rehabilitation medicine [13, 14, 15].

1.1 Motivation

Although VR HMDs are designed to provide an immersive environment where users feel they are part of, or inside a virtual environment, a major drawback of most immersive VR HMDs is that users cannot see and sometimes cannot hear their surrounding environment, partially becoming isolated from the real world [16, 17]. Even small tasks like finding the keyboard or picking up a phone call can become difficult when wearing a VR HMD [18, 19], and might lead to temporary isolation, where users are not able to get in touch with their friends and family while immersed in the VR space. One can argue that the mentioned difficulties can be avoided by having the users remove their HMDs, but such an interruption might have a negative effect on the VR experience or its outcomes, which might not be warranted, for instance, if a user just needs to check out a message or talk on the phone.

In the current world of social networks, users have a strong urge to be in touch with their friends and family through social media. According to statistics published in [20], more than 2.3 billion people are active social media users. Among them, 1.9 billion users engage with social media using a mobile device. In the absence of communication software built inside the VR, each time a user needs to use the smartphone to get in touch with friends and family or check for messages, the user needs to remove the HMD, which might deteriorate the VR experience. Through Mixed and Augmented Reality, additional devices such as an RGB camera attached to the HMD can be used as a window to the real world. However, finding the balance of Real and Virtual imagery that must be shown to the users is still hard, and it might still be difficult to read and perceive the text displayed on a smartphone when it is captured through an RGB camera and rendered through a commercial grade VR HMD. In this research, we will focus on the development of a system to allow users to interact with their smartphones within the VR world while wearing an HMD.

1.2 Methodology

To address the temporary social isolation of the users from the real world while wearing a VR HMD, we present a method that enables users to interact with their smartphone devices so that they can read messages, place calls, and use other apps while still being immersed in the VR environment. Users would no longer have to remove their HMD's each time they need to use their smartphone. To provide a window to the smartphone, our approach uses an infrared (IR) camera in the form of the Leap Motion device [21]. The images captured from the Leap Motion device are processed to detect the presence of the smartphone in the real-world. Once the smartphone is accurately detected in the real world, the scene in the VR world is augmented with the view of the user's smartphone, therefore enabling the users to use their smartphone without removing the HMD.

The first step to augment the scene inside the VR HMD with a view of the user's device is to accurately detect the edges of the smartphone in real-time. A smartphone is a texture-less object, in the sense that it lacks any repeatable or constant image patterns on its screen and/or surface, that can be used by existing feature extraction approaches to detect and track it. As feature extraction approaches like SIFT [22], SURF [23] or ORB [24] could not be used, we decided to explore edge-based object detection to detect the smartphone in real-time and designed an algorithm using some basic image processing. Our initial algorithm was solely based on image processing techniques and could successfully detect the smartphone edges in the Leap Motion images but with a high number of incorrect detections (high false positive rate). We then decided to explore the use of machine learning approaches to improve the specificity of our system while maintaining a high recall rate [25, 26]. As the smartphone detection must be done in real-time and the detection system should not deteriorate the user VR experience (e.g., by slowing the response of the VR environment), the machine learning approaches considered had to generate models that could be embedded into the detection system without a lot of computational overhead. Thus, we assessed the performance of three statistical learning methods that met these requirements: Logistic Regression (LR) [27], Linear Discriminant Analysis (LDA) [28] and Quadratic Discriminant Analysis (QDA) [29]. A justification of why the above mentioned three statistical classifiers were considered to improve the specificity of our system is given in 3.3.1.

In sum, our new device detection system first processes the images obtained from the Leap Motion device using standard image processing techniques for smartphone detection. Then, we identify and extract several unique features (attributes) from the processed image, and gave these features to the statistical model to discriminate between correctly or incorrectly detected smartphone edges in the image. Based on several performance measurements we decided to integrate the QDA model into the smartphone detection system. Once the new smartphone detection system successfully identifies the smartphone edges in the input image, the VR scene is augmented with the view of the user's smartphone screen so that the user can interact with the device without removing the headset.

1.3 Outline

The Thesis is organized as follows: Chapter 2 summarizes the previous work done in Augmented Virtuality, texture-less object detection and tracking, and previous instances of use of machine learning methods in image processing. Chapter 3 describes the system setup, devices used, and applications required to make the system work. It also describes the system design and implementation of the image-based smartphone detection system (IBSD), and the smartphone detection system using statistical classifiers (SDSC). Chapter 4 discusses our achievement, presents results and describes limitations of our system. Chapter 5 lists future enhancements and summarizes our findings.

Chapter 2

Background

2.1 Never Blind in VR: Augmented Virtuality to the rescue

While wearing an HMD, interacting with real world objects is a challenge, as users are partially isolated from the outside environment. Users cannot perceive most of their environment because their sight is occluded by the HMD and sometimes they cannot even hear their surrounding environment if they are wearing headphones inside the VR [16]. Under these conditions, interacting with other people and finding objects like desktops and peripherals or a cup of coffee is difficult [18, 19]. Milgram et al. [30] defined the term 'Reality-Virtuality (RV) Continuum', which states that the real environment and the virtual environment are the two ends of a continuum and should not be considered as an alternative to each other. One end of the spectrum represents the real world which surrounds the user and at the other end of the spectrum lies the virtual environment which is computer generated. As we move along the spectrum, either computer generated models are embedded into the real environment or elements of the real world are embedded into the virtual world. One point along this spectrum gives rise to a term called ‘Augmented Virtuality’, where elements of the real world are embedded into the virtual environment [31]. This differs from ‘Augmented Reality’ where virtual images are embedded inside a real world scene. To address the partial isolation issue, solutions usually involve the use of ‘Augmented Virtuality’. These solutions embed user’s hands and/or objects that are in close proximity to the user’s interaction space with the help of a head-mounted RGB camera, a head-mounted RGBD camera, or a external depth sensing device like Microsofts Kinect [32].

Among the early work that involved combination of real and virtual world elements in a scene was the SIMNET project [19]. It could overlay real-world objects onto a virtual world environment, as well as overlay virtual world objects onto a real world environment. With the help of a head-mounted camera placed on an HMD, the proposed system was able to provide a window to the outside world from within the virtual environment. This enabled users to perceive real world objects within the virtual world. In 2004, Regenbrecht et al. [33] described an Augmented Virtuality-based videoconferencing system that consists of HMD and an additional RGB camera. The RGB camera was used to capture video images of the user and track user’s head at the same time. The captured video was then embedded as a 2D video-plane into a virtual 3D space to give users an impression of a videoconference.

A more recent solution that involved the use of 'Augmented Virtuality' was presented in 2009 by Steinicke et al. [34]. The solution presented was able to model a user's hands and body inside the virtual environment. It was achieved with the help of a head-mounted RGB camera with a resolution of 640 x 480. Infrared LEDs were also mounted on top of the HMD to track the user's position within the physical real-world. In 2014, Tecchia et al. [35] and Ha et al. [36] demonstrated a system that allows the users to see and use their hands and body while interacting with virtual objects placed around the virtual environment. The demonstrated system consists of an RGBD camera that captures users hands in real-time and embeds them in the virtual world. The use of an RGBD camera allows the system to capture a textured depth image of hands and body which can easily be embedded in the virtual scene. In 2015, McGill et al. [18] presented a user study to help identify usability challenges faced by the HMD users. The study involved the use of an external head-mounted camera and chroma-key image segmentation to embed peripheral devices like the keyboard in the virtual environment to examine users' ability to interact with real world objects. The user study concluded that there is a need to embed aspects of reality into virtual environments to enhance the interactivity of users with peripheral devices. A similar implementation that uses an additional head-mounted RGB camera and chroma-keying technique to segment the captured image into foreground and background has been described in [37, 38, 39, 40].

Other researchers have been able to address other aspects of the isolation issue using Microsoft's Kinect, a depth sensing camera which can provide visual feedback of the objects that surround the user. The Kinect allows room-wide sensing capabilities, which makes it ideal for user tracking and gesture identification. One such system was presented by Dassault Systémes [41]. The system presented was successfully able to embed 3D point clouds obtained by the Kinect V2 in the user's virtual space and display elements of the real world that are in close proximity to the user. Although the system was not portable, they were able to display point clouds that represent people present in the same room, allowing people to see each other in the virtual world and throw and catch virtual objects at each other, partially addressing the social isolation issue. Another research conducted by [18] that involved a study of engagement dependent awareness of other people present in the same room was introduced in 2015. Similar to the system presented by Dassault Systémes, the authors used the Kinect to insert other people's silhouettes whenever they are within a close proximity to the user. In order to make it engagement-dependent, the other people's outline is initially faded in the VR world and then if the user wishes to engage, the users became visible in full color.

So far, most of the research done in this area has focused on how to make the user aware of other users or objects present in close proximity, and little research has been done to help the user stay in touch with other parts of the real world which are not in close physical proximity. The main motivation behind this research is to allow people to stay in touch with others and their digital life by allowing them to use their personal computing and communication devices, such as smartphones and tablets, from inside the VR world without removing their HMD headset.

2.2 Texture-less object detection and tracking

Existing feature-based object detection and tracking approaches like SIFT [22], SURF [23] or ORB [24], rely on matching descriptor features like blobs, corners or patterns in texture, but may not be effective in detecting featureless objects [42, 43]. As a smartphone device is a texture-less object, in the sense that it lacks any repeatable and constant image patterns on its screen and/or surface that can be used to detect and track it using existing feature-based techniques, feature-based approaches like SIFT, SURF or ORB could not be used. Thus, we explored the use of texture-less object detection techniques.

Most of the state-of-the-art techniques present in the area of texture-less detection use edge-based template matching [44, 45, 46, 47, 42]. The earliest record of texture-less 3D object detection based on template matching was presented by Olson and Huttenlocher [48] in 1997. The technique presented by them consists of representing the target 3D objects by edge maps of 2D views (from every possible viewing angle) of the object. These 2D edge maps along with a Hausdorff measure [49] that incorporates both location and orientation information are then used to determine the position of the target object in the target environment. Although effective, the data size required to correctly recognize and estimate the pose of the target object was large. Similarly in 2002, Steger's [46] method for industrial inspection matched 2D views of the target 3D object by using a robust similarity measure. Although the similarity measure was robust with respect to illumination changes and occlusion, when thousands of templates were used (in case of 3D objects for each and every pose) its speed decreased, as demonstrated in [45]. In 2010, Hinterstoisser et al. [44] presented a new template method called 'Dominant Orientation Template' based on local dominant gradient orientations. The new method introduced was robust with respect to small image transformations and could easily handle texture-less objects. Although fast evaluations of templates were obtained by using binary operators, the results varied with respect to template size and slowed down while handling multiple objects in an image.

Other researchers have explored texture-less object detection by combining template matching with a particle filter framework. One such system was presented in 2007 by Lee et al. [50]. It uses a sequence of images captured at different orientations and poses to recognize and estimate the pose of a texture-less 3D object. They then used a novel particle filter based probabilistic method for recognizing the target object and estimating the pose from the saved sequence of images with the help of template matching. In 2012, Choi and Christensen [42] presented a texture-less object detection method that uses an efficient chamfer matching algorithm [51] to match 2D edge templates and the query image. Using chamfer matching, they first estimated a set of course pose hypothesis of the target object, then initialized a particle filter framework with the result set and kept refining the pose of the target 3D object from subsequent images to improve the correspondence between edge points in the generated object model and the input image. In addition, others have utilized 3D CAD models instead of physical real objects to generate templates for template matching algorithms, like in [47]. These edge-based template matching techniques perform better than feature-based technique for texture-less object detection; however, they suffer from problems related to occlusion, cluttering and scalability of template library [43]. While some researchers have been able to reduce the search time required for template based matching such as in [52, 53, 54] the underlying problems related to occlusion, cluttering and scalability still remain.

In [43], Tombari et al. took a complete different approach, where instead of matching a template from the model template library they rely on descriptor-based object detector. In this method, a descriptor called BOLD (Bunch of Lines Descriptor) is proposed that can be injected into the standard SIFT-based object detection pipeline to provide performance improvement. The proposed method first extracts BOLD descriptors from the objects present in the model library and matches them with the BOLD descriptor extracted from the input image by using Euclidean distance [55] and FLANN Randomized Kd-tree forest [56] algorithms. Although the proposed method has impressive detection rates, the method is limited in detecting simple objects (made out of few lines) in scenes with heavy occlusion and clutter. Similar to [43] for descriptor-based object detection, Chen et al. [57] presented a detector termed BORDER (Bounding Oriented-Rectangle Descriptors for Enclosed Regions) to recognize texture-less objects in highly clutter and occluded images. BORDER follows a SIFT-like three step pipeline that consists of detection of interest-points by means of linelets (elongated line-segments divided into smaller equal-sized fragments), followed by a descriptor based on oriented rectangles, and then matching based on 'point-of-view' scheme. Although the method described is good at detecting texture-less objects, the runtime of BORDER based detector is high compared to the descriptor proposed in the BOLD-based descriptor.

In our case, as smartphones may vary in size and aspect ratio, existing template-matching, texture-less object detection methods may require a large template database. Ideally, the template database would consist of records for every possible 3D position of the smartphone as well as records for each variation of aspect ratio of the smartphone making the template database large. Searching such a large template database can prove to be inefficient for real-time tracking. In addition, creating an initial large edge-based template database or a set of BOLD descriptors with each and every possible smartphone currently available is highly impractical.Therefore, we decided to make use of traditional edge based object detection method as described in Chapter 3.2.

2.3 Machine learning in image processing and HCI

The edge based object detection method devised by us, although robust in locating the presence of smartphone edges within the input image, at times did not produce accurate results. Due to illumination changes, occlusion and clutter, there were some instances where edges of objects surrounding the smartphone were incorrectly included as part of the smartphone. For augmenting an object in VR environment it is important to detect and localize the target object in the input image accurately.

To address the temporary social isolation of the users from the real world, we needed a reliable method that would enables users to interact with their smartphone devices so that they can read messages, place calls, and use other apps while still being immersed in the VR environment. A detected false smartphone edge would not only produce undesirable results while augmenting the smartphone device but might also place the user's fingers in a false position in the VR scene, making it difficult for the users to use their smartphone. We need to discard those cases where edges of other objects were considered to be part of the smartphone. We basically needed a method that would identify whether or not the detected edges are part of the smartphone. As machine learning classifiers do this reliably and also run in real-time, we considered machine learning approaches that would lower the false detection rate and make the image-based object detection method reliable, without adversely affecting its real-time execution.

An early case of the use of machine learning in object detection was presented by Schneiderman and Kanade [25], where a statistical model was used within a face detector to account for variation in orientation and pose of the person's face. In order to cope with variation in pose estimation of a face, they followed a two-part strategy. First, they used a view-based detector for each specific orientation (e.g. right profile of face), then, they trained a statistical model within each detector to classify whether or not the detected object belongs to a non-object (objects other than face profile). By combining a statistical model with view-based multi detectors they were able to achieve an accuracy of 92.8%. In 2001, Viola and Jones [58] described a method based on machine learning for fast object detection in an image. They used a modified version of AdaBoost [59], which boosts the classification performance of simple classifiers to form an efficient classifier. The cascaded classifier was able to achieve highly accurate detection rate and perform 15 times faster than any previous object detection algorithm at the time.

In [26], Takeuchi et al. proposed a novel method for vehicle detection and tracking by using a combination of latent support vector machine (LSVM) [60] and Histograms of oriented gradients (HOG) [61] on previously defined deformable object model-based detection algorithm [62]. LSVM was used to reduce the false negative detection rate while maintaining the false positive rate obtained from deformable object model-based detection algorithm. By combining LSVM with HOG, they were able to achieve an average vehicle detection rate of 98% and an average vehicle tracking rate of 87%. Recently in 2014, Zhang et al. [63] proposed a human posture recognition system that uses SVM to identify pre-defined human postures from depth images. The depth image captured by Microsoft's Kinect is used to infer skeleton information represented by nine features. These extracted features are then fed to the SVM to identify the pre-defined postures from input images. The proposed method was able to achieve an overall accuracy of 99.14%.

Similarly in 2015, Marqués and Basterretxea [64] developed a hand gesture recognition system that uses information from an accelerometer attached to the fingers, to dynamically recognize seven different gestures. The researchers compared performance of three classifiers: Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and Extreme Learning Machines (ELMs). The experimental results obtained by them show that an accuracy of 98% can be achieved by using ANNs or ELMs. In 2015, Hai et al. [65] used ANNs and K-Nearest Neighbors (K-NNs) for facial expression classification. Using a combined proposed method 'ANN_KNN', they were able to classify seven basic facial expressions in categories such as anger, fear, surprise, sad, happy, disgust and neutral. The proposed method was evaluated on JAFEE database [66] and a classification precision of 92.38% was obtained.

Machine learning has also been extensively used in medical applications that involve processing of images captured from various medical instruments to identify a disease. For example, in 2015 Moreira et al [67] proposed a method that aims at developing an upper body evaluation method for women's who have undergone breast cancer treatments. To detect lymphedema caused by removal of axillary lymph nodes, they used Microsoft's Kinect to extract features from upper-body limbs motion and used these features to train LDA, NaiveBayes and SVM classifiers. The classifiers were then compared to find the best performing classifier. The results published by the team indicate that SVM was the best classier with 19% miscalculation error. While in the same year, Nayak et al. [68] proposed a system to classify brain's magnetic resonance images (MRIs) into normal or pathological brain. The proposed methods first extracts features from MR images by utilizing two-dimensional discrete wavelet transform, then reduces the extracted features by a significant number using principal component analysis (PCA) and linear discriminant analysis (LDA) algorithm. The reduced significant features are then applied to a random forests classifier to determine if the brain is normal or pathologic. They used a standard dataset 'Dataset-255' introduced by Das et al. [69] to evaluate the proposed method and were able to obtain a classification accuracy of 99.22%.

To consider the integration of a classifier into our traditional edge-based object detection algorithm, we needed a classifier that could satisfy constraints such as detection of smartphone in real-time, ease of integration with the existing solution, no additional process calls to external programs or libraries that might slow down the smartphone detector, and no additional storage of instance vectors (as required by instance-based classifiers such as K-NN [70]). Considering the constraints, we decided to evaluate three statistical classifiers: Logistic Regression (LR) [27], Linear Discriminant Analysis (LDA) [28], and Quadratic Discriminant Analysis (QDA) [29].

Chapter 3

System Design and Setup

This chapter first lists out the hardware used for our system setup to test out the algorithm designed. It then provides details about various steps used in Image-Based Smartphone Detector (IBSD) and explains the problem faced by the designed algorithm. This chapter then explains the need for statistical classifier, the steps required to extract features, assessment of the classifiers and creation of mathematical model from the analyzed data.

3.1 System Setup

To setup the system that would allow testing of the designed algorithm, the following components are required:

1. **Oculus Rift enabled desktop PC**

   A desktop PC that will support the Oculus Rift DK2 device. The desktop PC

Figure 3.1: Oculus Rift DK2 Device

we used has the following specifications: 24 GB RAM, two Nvidia GTX 670 graphics cards each with 2 GB memory, Intel i7 processor, HDMI video output port, four USB 3.0 ports and Windows 10 professional edition installed on it. The recommended system specifications for the Oculus Rift DK2 can be found in [71].

2. **Oculus Rift DK2 device**

   A virtual reality headset offered by Oculus to developers to build and test VR applications and games. The Oculus Rift DK2 device is shown in Figure 3.1. The Oculus Rift DK2 device specification includes: a display resolution of 960 x 1080 pixels per eye, refresh rate of up to 75 Hz and 100° field of view. The Oculus Rift is connected to the desktop PC by using one HDMI port for video display and one USB 3.0 port for power requirement as shown in Figure 3.2.

3. **Leap Motion VR device**

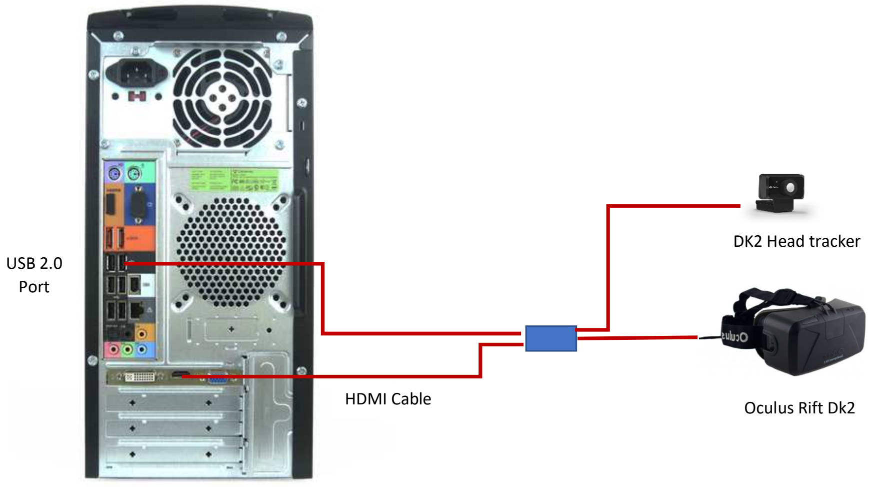


Figure 3.2: Oculus Rift DK2's connections to desktop PC

An IR motion sensor device that captures hand gestures and provide hand tracking functionality inside the virtual reality space. The Leap Motion VR device, as shown in Figure 3.3, was used as a camera that would act as a window to the real world inside the VR environment. The Leap Motion VR device operates at 20-200 frames per second depending on available computing resources, activity within the device field of view, and software tracking settings. The Leap motion API can be used to capture an IR image of resolution 640 x 320 from the Leap motion device.

4. **Leap Motion's Universal VR Dev Mount**

   A mount that can be attached to the Oculus Rift DK2 device to hold the Leap

Figure 3.3: Leap Motion VR Controller

Figure 3.4: Leap Motion's Universal VR Dev mount

Motion VR device in such a way that the IR sensors are facing away from the user. The Universal VR Dev mount setup is shown in Figure 3.4, while Figure 3.5 shows the arrangement after the Leap Motion was attached to the Oculus Rift DK2.

5. **A Smartphone Device**

A Smartphone device with Android 4.2 (Jelly Bean) or higher that has basic sensors such as accelerometer and gyroscope and internet connectivity is required. The Smartphone device used for experimentation was 'One Plus One' with 3 GB of RAM, Qualcomm Snapdragon 801 Quad-core processor, 5.5 inches

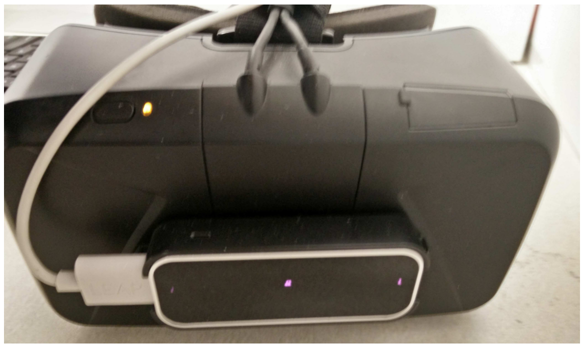


Figure 3.5: Leap Motion VR device attached to Oculus Rift DK2

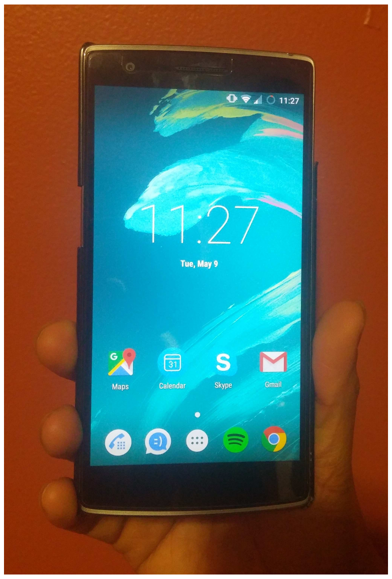
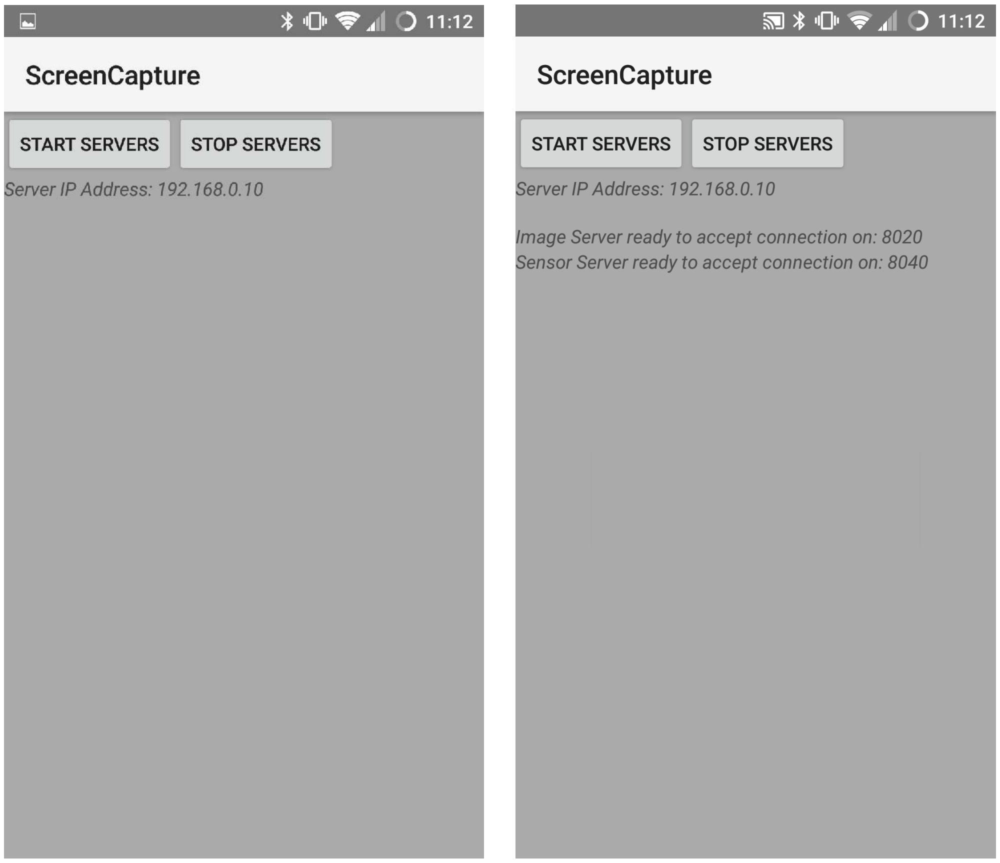


Figure 3.6: The smartphone device used for testing IBSD: OnePlus One

of display and 1080 x 1920 pixels (401 PPI pixel density) screen resolution with Android 6.0 (Marshmallow) operating system. The Smartphone device used is shown in Figure 3.6

6. **Android Application to Transfer Smartphone's Screen**

An Android application that continuously captures the screenshots of the cur-



Server IP Address: 192.168.0.10

Server IP Address: 192.168.0.10

Image Server ready to accept connection on: 8020
Sensor Server ready to accept connection on: 8040

(a) (b)

Figure 3.7: Android application: (a)Screen shown when the app starts. Note the app displays the TCP/IP server's IP address on the screen (b) Screen shown after User clicks on 'Start Servers' button

rent contents displayed on the Smartphone's Screen and, using TCP/IP protocol transfers the captured screenshots to the designed VR application, was created using Android Studio [72]. The rate at which the Android application captures the screenshots was 33 frames per second. In addition to the screenshots, the Android application was also designed to send the device's orientation and position change information from the accelerometer and gyroscope sensors. The sampling rate of the gyroscope and accelerometer obtained for OnePlusOne device was 5Hz and 15Hz, respectively. The Android application was designed as a TCP/IP server and would initiate continuous transfer of screenshots and sensor data to any client connected to it. As shown in Figure 3.7, it has two buttons that are used to start or stop the TCP/IP server. Once the TCP/IP server is initialized, it displays the IP address and port it is bound to. The displayed address is used by TCP/IP client (VR application) to connect to the server and receive the captured screenshots.

7. **VR application**

   A VR application was created using Unity [73] to display the VR environment using the Oculus Rift DK2 device. The Unity version 5.2.2 was used for development. Unity 5.2.2 provides an extensive support for Oculus Rift DK2, which was removed from the latest versions of Unity. Unity 5.2.2 can be downloaded from [74]. The VR application designed in Unity displays cubes of different colors to users; these cubes are placed at random locations within the VR environment. Figure 3.8 shows the designed VR application.

   The Leap Motion VR device was integrated with Unity using Unity assets provided by Leap motion. The Unity assets can be downloaded from [75]. Leap Motion's Unity assets provide the relevant code and objects inside Unity to

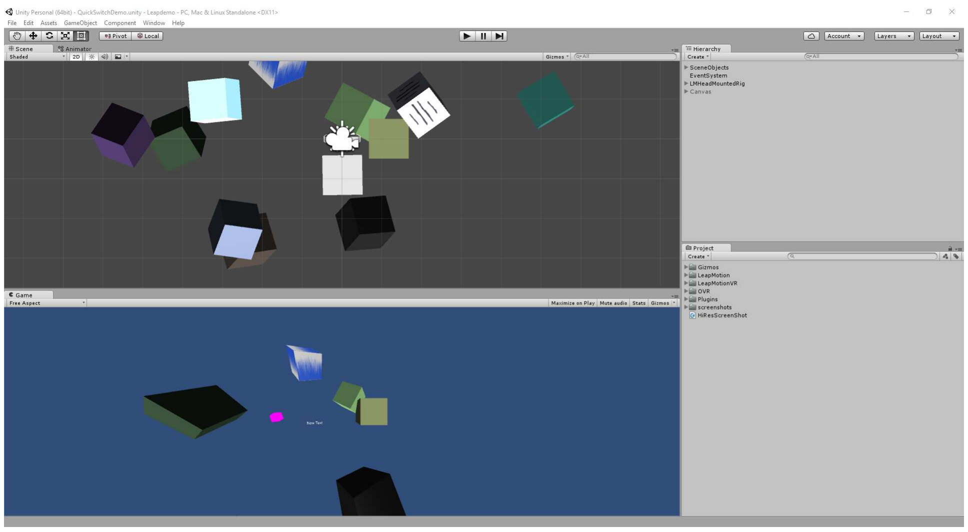


Figure 3.8: VR application created in Unity to display cubes of different colors to users. The cubes are placed at random locations within the VR environment

display images captured by the Leap Motion device. The images captured by Leap Motion basically are blended with existing VR objects (cubes in our application) and are displayed in the background of the VR space in such a way that it covers the entire user's VR visible space.

The Image-Based Smartphone Detector (IBSD) explained in section 3.2 was integrated inside the Unity application to detect the presence of the smartphone in the images captured by Leap Motion. A Unity's quad object [76] called 'Smartphone Quad' was used to display the screenshot captured and transferred from the Android application. The position and dimensions of the 'Smartphone Quad' were updated dynamically depending on the position of the smartphone tracked in real world. Another quad object (Finger Quad) was used to display the outline of the user's fingers. The Finger Quad was placed on top of the Smartphone Quad and the images displayed in both the quads (Smartphone's screenshot and User's Fingers) were blended so that a user could visualize the position of their fingers while operating the Smartphone. The application designed also acts like a TCP/IP client that connects to the Android application and receives the smartphone's screenshots.

8. **Local LAN setup**

   To enable the VR application to connect to the TCP/IP server running on the Android smartphone, a local LAN setup is recommended. A local LAN setup will not only ensure communication between the smartphone and VR application but will also enable transfer of screenshots captured from smartphone at a faster rate due to reduced communication latencies. We used a wireless D-Link router with data transfer rate of 54 Mbps and frequency band of 2.4 GHz.

9. **Cubicle to test IBSD**

   To test the smartphone detection algorithm, we set up a cubicle as shown in Figure 3.9 to limit the interference from other objects surrounding the user while testing IBSD. The cubicle dimensions recorded were 22 inch height, 24 inch width and 20 inch length. Three sheets of white paper were used as the walls on three sides while the fourth side was kept open. All interactions with the smartphone were done inside the cubicle.

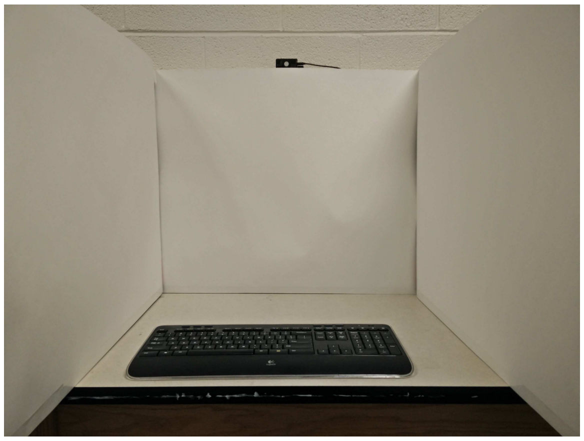


Figure 3.9: Cubicle to test IBSD

3.2 Image-Based Smartphone Detector (IBSD)

To correctly augment the scene inside the immersive VR environment with a view of the user's smartphone device, one needs to obtain in real-time an accurate identification of the edges of the smartphone or tablet using the images obtained from the LeapMotion sensor. Once the smartphone's edges are correctly identified, one can proceed with the augmentation logic to display the smartphone's screen and position of user's fingers inside the VR environment. To identify the smartphone's edges, a set of image processing steps (as described below) were devised that, when applied to the images captured by the Leap Motion VR device, would lead to an identification of the smartphone.

The steps involved in detecting a smartphone in an image captured by the Leap Motion device are outlined in Figure 3.10. All the image processing steps were performed using OpenCV [77]. OpenCV or Open Source Computer Vision is a set of

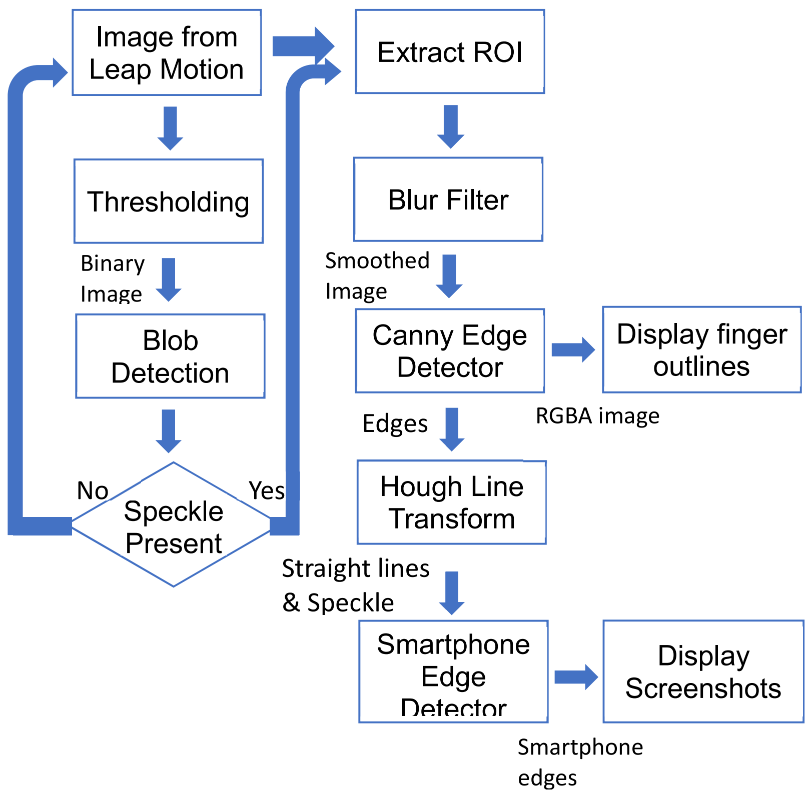


Figure 3.10: Flow diagram of the Image Based Smartphone Detector (IBSD)

libraries of programming functions for real-time computer vision algorithms. An OpenCV wrapper known as EmguCV [78] was used to access OpenCV API's in C# language. Detection of any object in an image involves two steps: locate the region of interest (localization) and then recognize if the desired object is present in that region or not (object detection).

1. **Localization of the Smartphone**

   In our case, instead of searching the entire image for the smartphone, we relied on a finding that helped us localize the smartphone: The Leap Motion device is basically an Infrared (IR) Camera that relies on the reflected IR light back to its

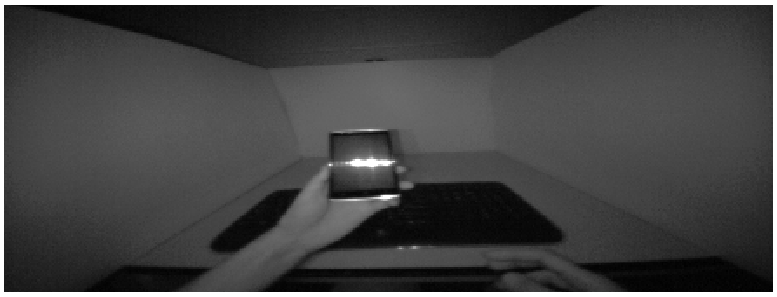


Figure 3.11: Image captured by leap motion device with speckle at the center

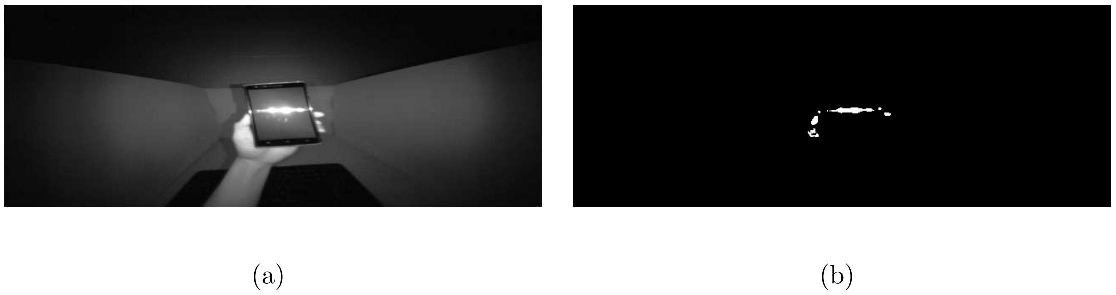


Figure 3.12: Binary Threshold. (a) Original image captured by Leap Motion device (b) Image obtained after applying a binary threshold

sensors to detect user's hands. We noticed that whenever an object which has a reflective surface such as the smartphone is held in front of the Leap Motion device, it produces a blob with high brightness (which we call smartphone's speckle) as seen in Figure 3.11. We used this property to localize the smartphone in the Leap Motion image. This only works under the assumption that users will position their smartphone in front of the HMD in such a way that they can read and/or view the smartphone's screen correctly, i.e., the smartphone's screen is facing the HMD.

For localization, once the VR application captures grayscale images from the

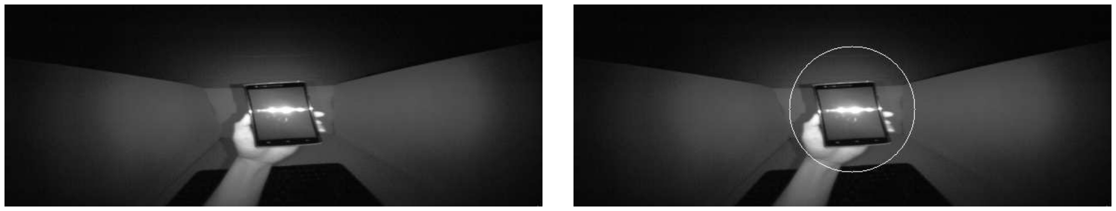
Leap Motion device through a set of API's provided by Leap Motion, we first apply a fixed-level threshold on the grayscale image to find the bright speckle in the image. A binary threshold value of 230 was used for binary thresholding [79]. To obtain a threshold value of 230, images obtained from Leap Motion were thresholded using different values of threshold and an optimal threshold (Threshold = 230) was selected such that only speckle will be visible in the image. A sample of binary threshold result is shown in Figure 3.12. A blob detection algorithm [80] is applied on the thresholded image to extract blobs present. The blobs are filtered by area such that small blobs with area less than 20 units and large blobs with area more than 250 units are discarded. The remaining blobs are sorted out by size and the blob with maximum size is selected as the speckle. The speckle's center and size are marked.

2. **Detection of the Smartphone**

   The smartphone detection algorithm works as follows:

   (a) Extract Region of Interest (ROI)

   Once the smartphone is localized, we extract a region of interest (ROI) from the original grayscale image obtained from the Leap Motion device. In our case, the ROI is a circular region of radius of 90 pixels around the speckle center, as shown in Figure 3.13. The radius of 90 pixels was decided after experimenting with various sizes of ROI. The selected ROI captures the smartphone irrespective of the position of speckle on the smartphone's



(a) (b)

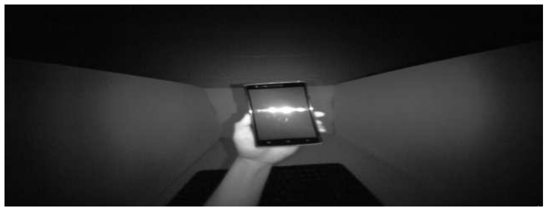
Figure 3.13: Extracting Region of Interest (ROI). (a) Original image captured by Leap Motion device (b) The circle denoted the ROI that will be extracted from the original image for further processing.

screen or the distance of smartphone from the HMD, i.e, if the speckle is detected correctly the ROI may contain the smartphone even if the smartphone is held near the HMD or as far as user's hand can possibly be far away from the HMD.

(b) Extract Straight Lines

After extracting the ROI, in order to obtain best results with any edge detection algorithm, we applied a normalized box blur filter (with kernel size = 3x3) [81] to smooth the extracted ROI image. The smoothed image is then sent to a Canny edge detector [82]. Figure 3.14 shows a sample image after Canny Edge Detection.

Current smartphones have a particular geometry, i.e., they are rectangular and have straight edges. We use a basic principle in perspective projection for detecting these straight smartphone's edges, i.e., in perspective projection, straight lines in 3D will project to straight lines in the 2D image.


(a)

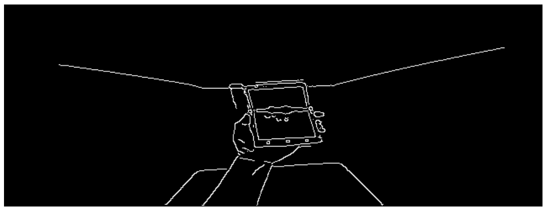
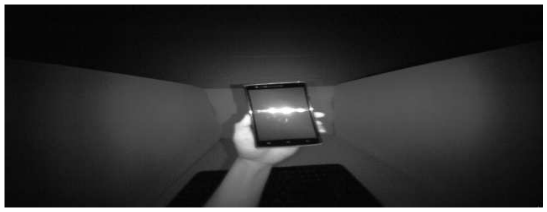

(b)

Figure 3.14: Canny Edge Detection. (a) Original image captured by Leap Motion device (b) Image obtained after applying canny edge detector


(a)

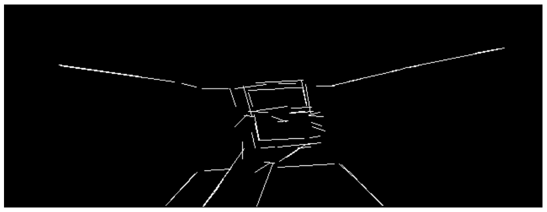

(b)

Figure 3.15: Probabilistic Hough Line Transform. (a) Original image captured by Leap Motion device (b) Image obtained after Hough Line Transform

We apply a Probabilistic Hough Line Transform [83] algorithm available in OpenCV to detect these straight lines. Each extracted straight line, in addition to the start and end point of line, also contains information about the orientation of the line with respect to the image's X axis. Figure 3.15 shows a sample image after Hough Line Transformation.

(c) Smartphone Edge Detector

Once the straight lines are obtained, we pass these lines (potential smartphone edges) through the smartphone edge detector module. The smart-

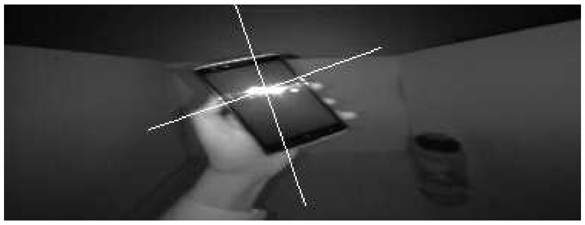
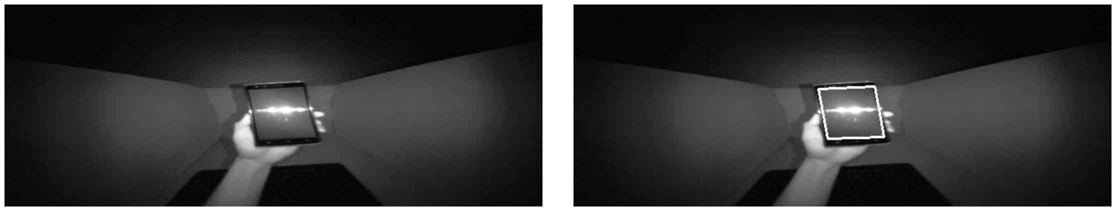


Figure 3.16: Image captured by leap motion device with speckle at the center of the cross hairs

phone edge detector module then applies geometrical constraints on the extracted ROI image to detect if a potential smartphone edge belongs to the smartphone or not. The steps followed by Smartphone edge detector are as follows: With the smartphone's speckle as the center, the smartphone detector module first shoots four straight lines (Edge Detector Lines) in four different directions (left, right, top and bottom of speckle) with respect to the current orientation of the smartphone, as shown in Figure 3.16.

The module then extracts only those potential smartphone edges that intersect with the Edge Detector Lines and place them in groups (LeftGrp, RightGrp, TopGrp and BottomGrp). The edges from LeftGrp and RightGrp are then compared to find pairs of edges that are nearly parallel to each other (the difference between the orientation of both the edges is minimum). Similarly, the edges from TopGrp and BottomGrp are compared to find pairs of edges that are nearly parallel to each other. The two pair obtained (left-right and top-bottom) are considered as smartphone



(a) (b)

Figure 3.17: Extracted Smartphone Edges. (a) Original image captured by the Leap Motion device (b) Extracted smartphone edges drawn on top the original image

edges. Figure 3.17 shows an example of the pair of edges extracted by the Smartphone Edge Detector that are considered to be smartphone edges.

3. **Display smartphone screenshots**

   Once the smartphone edges are detected, the 'Smartphone's Quad' object (Unity's quad object) in VR application gets modified with any changes in the edges of smartphone. The screenshots transferred from the Android's TCP/IP server are rendered on to the Quad to create an overlaying effect, such that the screenshot would match the position and shape of the smartphone present in the image captured by Leap Motion device. Figure 3.18 shows one such example after a successful detection of smartphone edges, the Quad object is rendered with the screenshot of smartphone's screen and overlaid on the leap motion image.

4. **Display Finger outlines**

   The problem with overlaying the smartphone's screenshot on top of the actual

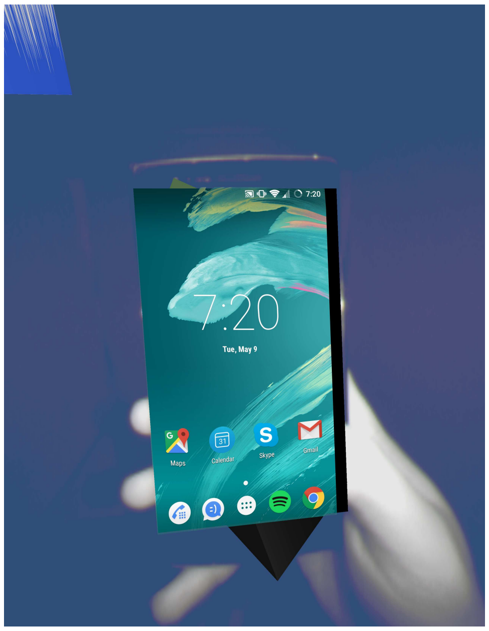
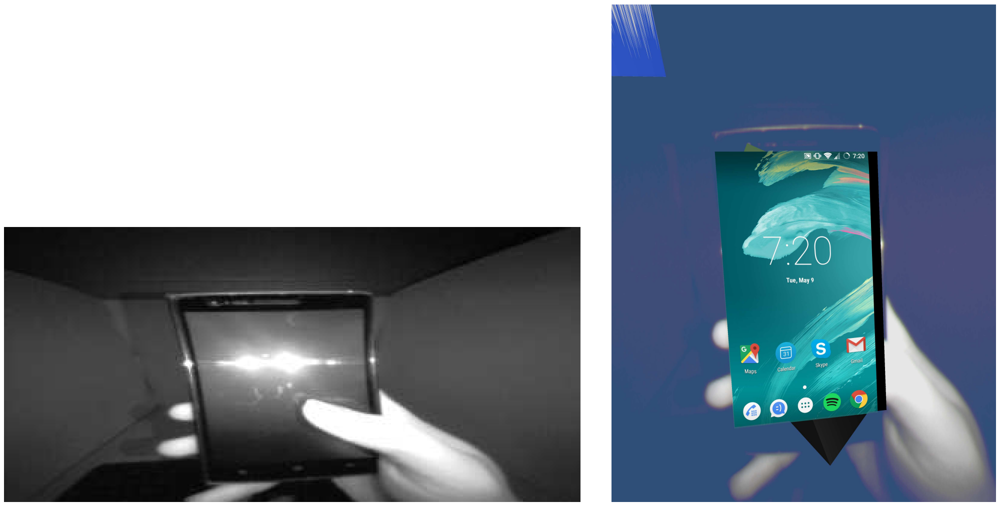


Figure 3.18: Smartphone's screenshot overlaid on the leap motion image. Note: Only a portion of the VR space is shown in the image

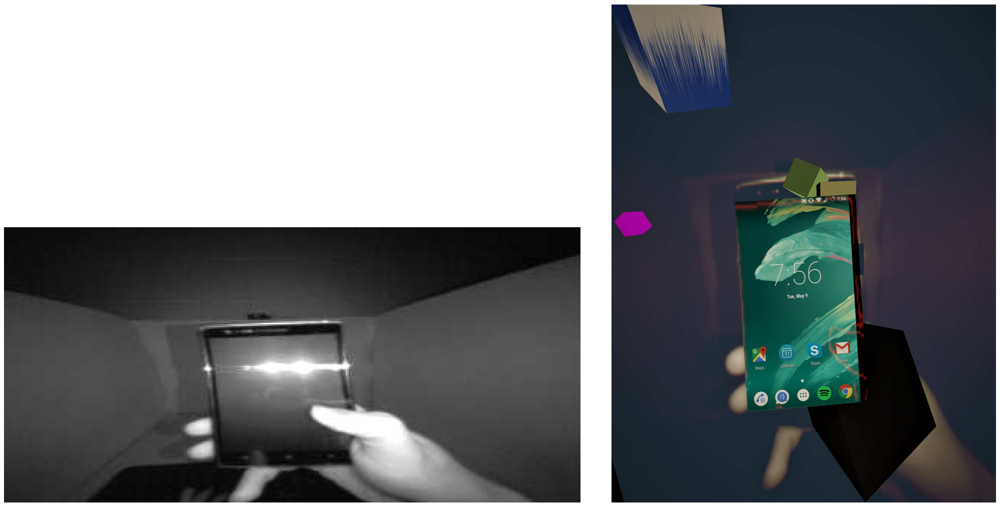
smartphone in the images captured by the Leap Motion device, is that the overlay obstructs the view of user's finger position as shown in Figure 3.19. To avoid the obstruction and let the user's view the position of their fingers to operate the smartphone, we did some image processing on the edges detected by the Canny Edge Detector [82]. The resulting edge image from Canny was first converted to an RGB image with all edges marked in red color. Alpha channel was added to the new image. The alpha channel contained grayscale values from the image captured by leap motion. This arrangement gives a transparency to the RGB image such that only the detected edges will be visible to the user, and the rest of the image becomes transparent. The final result can be shown in Figure 3.20.



(a) (b)

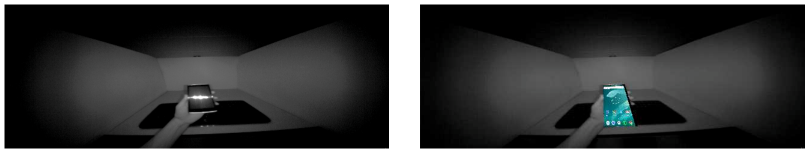
Figure 3.19: Smartphone screenshot overlay obstructs the view of user's fingers: a) The original image captured from leap motion b) the original image is mapped to the background in Unity and then overlaid with smartphone's screenshot

IBSD was able to detect the smartphone's edges; however, there were many instances where edges of other objects in the surroundings were also (mistakenly) considered as a smartphone edge. In short, our devised algorithm had a high false positive rate, which was unacceptable. Figure 3.21 shows instances where edges of other objects in the surroundings were considered to be edges of the smartphone. A detected false smartphone edge will not only produce undesirable results while augmenting the smartphone device in VR environment, but might also place the user's fingers in a false position in the VR scene, making it difficult for the users to use their smartphone.



(a) (b)

Figure 3.20: The resulting RGBA image overlaid on top of smartphone screenshot to show user's finger. a) original image captured from leap motion b) Image captured in Unity that shows user's finger in red outline overlaid on top of smartphone's screenshot.



(a) (b)

Figure 3.21: Edges of other objects in the surrounding are considered as a smartphone edge. (a) Original image captured by the Leap Motion device (b) The edge of the keyboard is erroneously considered to be a smartphone edge

3.3 Smartphone Detection with Statistical Classifier (SDSC)

After analyzing and testing IBSD (testing is discussed in Section 4.1), we found that IBSD was able to detect a smartphone in nearly 70% of the cases successfully; however, the remaining 30% were false positive instances where edges of other objects in the surroundings were also considered a smartphone edge. To lower the false positive rate and obtain better detection results, we considered the use of machine learning approaches, such as statistical classifiers [84], that would be able to determine whether or not the detected edges are indeed the smartphone's edges. To obtain better detection results, we compared three statistical classifiers: Logistic Regression (LR) [27], Linear Discriminant Analysis (LDA) [28], and Quadratic Discriminant Analysis (QDA) [29].

3.3.1 Justification of Statistical Classifiers chosen

To consider integration of a classifier into the exciting IBSD algorithm, we needed a classifier that could satisfy constraints such as detection of smartphone in real-time, ease of integration with the existing solution (possibly by some mathematical model), no additional process calls to external programs or libraries that might slow down the smartphone detector, and no additional storage of instance vectors (as required by instance-based classifiers such as K-NN [70]). The three statistical classifiers chosen: Logistic Regression (LR) [27], Linear Discriminant Analysis (LDA) [28], and Quadratic Discriminant Analysis (QDA) [29], each generate a mathematical model that can be easily integrated into the existing system, can evaluate the input and produce results in real-time, and do not cause significant computing overhead. In addition, evaluation of the mathematical models generated by these three statistical classifiers does not depend on any external program or software libraries which reduces the overall computing overhead. Also, as these classifiers do not depend on the any additional vector instances that were used for training them, additional memory requirement is minimal.

3.3.2 Feature extraction

To select the most suitable classifier, we assessed the predictive performance of these three statistical learning methods and selected the one with the lowest false positive rate at a comparable recall rate. Once the images captured by the Leap Motion device are processed by IBSD, we extract a set of 11 features from the detected smartphone edges. A list of the features extracted from the detected edges is given in Table 3.1, while Figure 3.22 illustrates these features in the processed images. Since the values of the extracted features are absolute X and Y coordinates of the smartphone detected in the image, these features are normalized. To normalize the features, the X coordinates of the four corners and X coordinate of the speckle are divided by the maximum width of an image, while the Y coordinates are divided by the maximum

Table 3.1: List of features extracted from detected edges

Feature (D)	Name	Description
1	SpeckleX	x coordinate of speckle in the captured image
2	SpeckleY	y coordinate of speckle
3	SpeckleSize	size of speckle
4	TopLeftX	x coordinate of top left corner of smartphone
5	TopLeftY	y coordinate of top left corner
6	TopRightX	x coordinate of top right corner
7	TopRightY	y coordinate of top right corner
8	BottomRightX	x coordinate of bottom right corner
9	BottomRightY	y coordinate of bottom right corner
10	BottomLeftX	x coordinate of bottom left corner
11	BottomLeftY	y coordinate of bottom left corner

height of an image. The maximum height and width of an image obtained from the Leap Motion VR device are 240 and 640 respectively. As the feature 'SpeckleSize' had no relationship with the image size, we used a different scale to normalize it. According to our empirical observations, we found that the maximum size a speckle can have was 40 pixels, so the 'SpeckleSize' feature is normalized by dividing its value by 40.

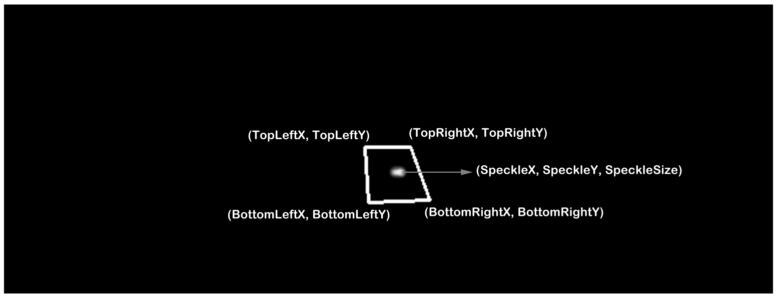

(TopLeftX, TopLeftY)
(TopRightX, TopRightY)
(SpeckleX, SpeckleY, SpeckleSize)
(BottomLeftX, BottomLeftY)
(BottomRightX, BottomRightY)

Figure 3.22: Features extracted from the detected smartphone

3.3.3 Assessment of Machine Learning Approaches

To create a Smartphone Detection with Statistical Classifier (SDSC) system, we first generated a data set and evaluated the performance of three classifiers using three different feature subsets. In this subsection, we describe the process followed to select the most suitable classifier for our application domain. The R programming language was used to perform most of the steps described below. The R code used for the analysis can be found in the file name 'SmartPhoneClassifier', located in an online repository at [85].

3.3.3.1 Dataset used

The dataset used to evaluate the performance of the statistical classifiers contained 312 instances or records. Each instance was made up of a set of 11 features (see Table 3.1) that were extracted from the smartphone's edges detected by IBSD as described in the previous section. The instances were classified into two groups, depending on whether the edges of the smartphone are correctly detected (group 1) or not (group

0). For each instance, a features vector along with its corresponding class is stored in the dataset. Out of 312 records, 222 (or 71.15%) records belonged to group 0, while 90 (or 28.85%) records belonged to group 1. The normalized dataset used can be found in a file named 'DatasetForSDSC', located in an online repository at [85].

3.3.3.2 Feature Selection

We undertook a feature selection process on the normalized data to select those features that allow the machine learning approaches to generate the models with the best predictive performance in terms of recall, false positive rate (1 - specificity) and Area under the ROC Curve (AUC). To estimate the performance measurements for each statistical learner (LR, LDA and QDA) and feature set, we carried out 10-fold cross-validation [86]. At the end of the 10-fold cross-validation, we obtained the average and standard deviation of each performance measurement. The classifier with the lowest false positive rate at a comparable recall was selected to be integrated into the SDSC implementation. The details on the classifier selection process are given in Section 4.2.2.

To select those features that are most relevant to construct a classification model, we created three different subsets of features. Each of the three features subsets were used to train LR, LDA and QDA as described in step 3.3.3.3. We compared the performance of each of the features subsets and used the subset of features that yield the best performing model to obtain the final model for integration with IBSD. The feature selection methods used are described below:

1. **Remove Redundant Features (RRF)**

   In this method of feature selection, we iteratively removed those features that were highly correlated with another feature. The 'findCorrelation' function in the Caret R package [87] was used to report the highly correlated features. A pair-wise absolute Pearson correlation coefficient of 0.7 was used as threshold to remove redundant features.

2. **Recursive Feature Elimination (RFE)**

   This is an automatic feature selection method that builds different models using Random Forest and identifies those features that are not relevant to construct an accurate statistical model [88]. We used the RFE implementation provided by the Caret R package.

3. **All available features**

   We also used all available 11 features from IBSD feature extraction for training the classifiers.

3.3.3.3 Training the classifiers

To evaluate the classification performance of LR, LDA, and QDA on our application domain, we performed 10-fold cross-validation on the dataset. We first randomized the dataset and divided it into ten folds (partitions) of roughly the same size. Each data partition had approximately 31 observations in it. For each fold, we left out one data partition for testing purpose and train LR, LDA, and QDA with the remaining data partitions. The model obtained during training is tested against the left out fold and performance measures such AUC, accuracy, specificity and sensitivity are recorded. The calculated performance measurements for all ten folds were then averaged out. After calculating the average performance measures for all ten folds, we compared the three classifiers. In the end, the classifier with the highest average AUC was selected.

3.3.3.4 Integration of the classification model with IBSD

Two of the constraints considered while selecting the classifiers to evaluate were: a) easy integration with IBSD and b) no additional process calls to external programs. To facilitate that, we used classifiers that can be represented with a mathematical model. The mathematical models for each classifier are given below:

1. Logistic Regression: For LR, the equation can be given as below [27]:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_{11} X_{11}}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_{11} X_{11}}} \tag{3.1}$$

   where $X = (X_1 \ldots X_{11})$ are features, $\beta_0 \ldots \beta_{11}$ are the learned parameters, and $p(X)$ is the predicted probability calculated for class 1. Observations are assigned to class 1 if $p(X) > T$, where T is a threshold set to maximize the classifier's performance.

2. Linear Discriminant Analysis: Assuming a Gaussian distribution, the Bayes

classifier can be approximated for LDA by the equation [28]:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k \tag{3.2}$$

where μ_k = mean of kth class, Σ = covariance matrix, π_k is prior probability of kth class, and $\delta_k(x)$ is the discriminant function for the class k. Observation x is assigned to the class for which $\delta_k(x)$ is largest.

3. Quadratic Discriminant Analysis: For QDA, the Bayes classifier can be approximated by the equation [29]:

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + log\pi_k \tag{3.3}$$

where μ_k = mean of kth class, Σ_k = covariance matrix for the kth class, π_k is prior probability of kth class and $\delta_k(x)$ is the discriminant function for the class k. Observation x is assigned to the class for which $\delta_k(x)$ is largest.

3.3.4 Comparing IBSD with SDSC

After integrating the best performing classifier with IBSD, we compared the performance of IBSD with that of SDSC. To do that, a dataset of 210 records was generated where each record contained three variables. Variable 1 was given a value of 1 to indicate a smartphone is identified by IBSD. Variable 2 was given a value of 0 if SDSC did not identify a smartphone and 1 if it identified a smartphone. Variable 3 indicated the actual class (or group) the record should belong to (0 - if the smartphone edges are not properly detected or 1 - if the smartphone edges are properly detected). Out of 210 records obtained, 59 (or 28.1%) records belonged to group 0 while 151 (or 71.9%) records belonged to group 1. The validation dataset used can be found in the file named 'ValidationDatasetForSDSC', located in an online repository at [85].

At the end of the 10-fold cross-validation, we obtained the average and standard deviation of each performance measurement. The classifier with the lowest false positive rate (< 0.2) at a comparable recall of 0.8 was QDA (Figure 4.4), thus, QDA was selected to be used in the SDSC implementation. QDA was then trained with the complete dataset to generate the classification model to be implemented. To do this, we obtained class means per feature and class covariances from the dataset. These values were then used to generate a mathematical model that was integrated with our existing smartphone detector.

3.4 Smartphone Detection with Generalized Statistical Classifier (SDGSC)

The data used for comparing the performance of LR, LDA and QDA classifiers above and training the best performing classifier (QDA) was limited to the data collected from the primary researcher of this thesis. It did not account for variations in sizes of the smartphone nor did it account for variations in the way different users use their smartphones at different positions and orientations. Therefore, although the initial results obtained from SDSC were promising, for users other than the primary researcher, there were some instances where SDSC would accept a false positive. The main limitation we observed after integrating SDSC was mis-alignment of the virtual smartphone panel with the real-world smartphone due to false positive instances accepted by SDSC. This caused jumpiness (constant position update) of the smartphone's display within the virtual environment even when the user held the smartphone in one position. To eliminate the jumpiness and improve the reliability of the designed system we decided to train the model using a large dataset collected from 10 different users.

Therefore, to train SDSC to accommodate for the different size of smartphones as well as different position in which a user can hold a smartphone, we undertook an additional data collection step where 10 users were invited to try out the designed VR system (IBSD) and data was collected anonymously during the event. The consent form, recruitment poster, and exit questionnaires asked after completion of the study can be located in Appendix A, B, and C respectively. The amount of time spent by each user on data collection was of about 15 minutes duration. The users were asked to use their personal smartphone device and they were instructed to perform the following steps given below:

1. Wear Oculus VR Head-mounted display
2. Pick up the smartphone placed on the table or handed to the user
3. Place the smartphone (with screen facing the HMD) in 5 different regions (center plus corners, clockwise from the top left) and at 3 different depths/distances for

each region designated in the VR space for a test run.

4. Place the smartphone (with screen facing the HMD) in 5 different regions (center plus corners) and at 3 different depths/distances for each region designated in the VR space for actual data collection.

5. The previous step was repeated with the phone in two basic orientations, analogous to the portrait and landscape printing orientations. At each designated position the user was requested to hold the smartphone in position for 2 seconds so that system will capture five images from the Leap motion VR device. Data points were extracted from these captured smartphone images and were recorded in a file.

6. Place the smartphone (with screen facing the HMD) at random locations anywhere in the VR space for the next five minutes. The system will continuously capture images from the Leap motion device, extract data points and record them in a separate file.

The different devices brought by users were OnePlus One [89], LG G4 [90], OnePlus 2 [91], LG G3 [92], HTC One M8 [93], LG Nexus 5 [94] and Moto G [95].

3.4.1 Dataset Used

The dataset collected by placing the smartphone in 5 different regions, at 3 different depths and two orientations (described in data collection step 4 and 5 above) for 10 users contained 3,000 instances or records (10 images at each designated position * 5 positions * 3 depths * 2 orientations * 10 users). Similar to the dataset used previously to train QDA 3.3.3.1, each instance in this dataset was made up of a set of 11 features (described in 3.1) that were extracted from the smartphone's edges detected by IBSD. The dataset collected by placing the smartphone at random location (described in data collection step 6) contained 5,600 instances or records. Each instance in this dataset was made up of a set of 11 features. Both the dataset (collected by designated and random placement) were combined to form a single dataset that contained 8,600 instances or records and the combined dataset was normalized by using steps described in 3.1. The instances in the combined dataset were classified into two groups: group 1 if the edges of smartphone are correctly detected or group 0 if not. Out of 8,600 records, 7,313 (or 85%) records belonged to group 0, while 1,287 (or 15%) records belonged to group 1. The normalized dataset used can be found in the file named 'DatasetForSDGSC', located in an online repository at [85].

3.4.2 Training the QDA classifier

A cross-validation for all three classifiers (LR, LDA and QDA) was performed using the same steps as described in 3.3.3.2 and 3.3.3.3 to verify if QDA was still the best performing classifier or not. Our results, as shown in 4.12, indicate that, although the overall accuracy was reduced by 3%, QDA still performed better than LR and LDA. After the evaluation, QDA was re-trained with the newly created dataset. To re-train the classifier, we once again obtained class means per feature and class co-variances from the dataset and then used these values to generate a mathematical model (by using Equation 3.3). The generated mathematical model was then integrated with our existing smartphone detector (IBSD) to obtain a new smartphone detector (SDGSC)

3.4.3 Comparing SDSC with SDGSC

After integrating the new generalized smartphone detector (SDGSC), we compared the performance of the previously obtained SDSC with that of SDGSC. To do that, a dataset of 1,200 records was obtained from a completely new user, i.e, a user who was not among the participant of the data collection study. Each record in the obtained dataset contained three variables (Variable1, Variable2 and Variable3). Variable 1 was given a value of 1 if SDSC classifies that the extracted edges belong to smartphone and 0 if not. Similarly, Variable 2 was given a value of 1 if SDGSC classifies that the extracted edges belong to smartphone and 0 if not. While, Variable 3 indicated the actual class (or group) the record should belong to 0 - if the smartphone edges are not properly detected, or 1 - if the smartphone edges are properly detected. Out of 1,200 records obtained, 1,070 (or 89%) records belonged to group 0 while 130 (or 11%) records belonged to group 1. The validation dataset used can be found in the file named 'ValidationDatasetForSDGSC', located in an online repository at [85].

3.5 Summary

To summarize, this chapter first lists out the hardware used to setup a system to test out the IBSD algorithm. It then provides details about various steps used in Image-Based Smartphone Detector (IBSD). Although IBSD was able to detect smartphone in a given Leap motion's image, there were many instances where the algorithm failed and incorrectly includes edges of surrounding objects as smartphone's edges. To lower this false detection rate, machine learning methods were applied. The chapter explains the modified detection algorithm 'SDSC' which is designed by combining IBSD and the best performing classifier. The steps required to find the best performing classifier are then explained. These steps include extracting features from Leap motion image, feature selection process, training the classifiers and creation of a mathematical model. Finally, the parameters on which the performance of IBSD and SDSC are compared are defined.

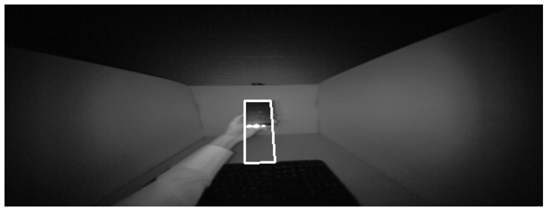
Chapter 4

Results and Discussion

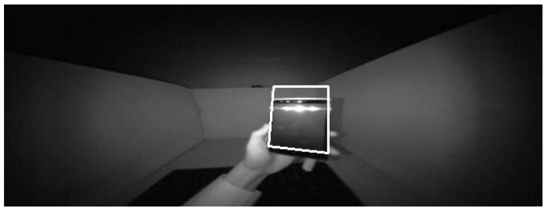
This chapter first lists out the results obtained by IBSD and discusses the drawbacks of the designed algorithms. It then discusses the results obtained in the process of selecting the beat-performing classifier. This chapter then analyzes the comparison between IBSD and SDSC. Finally, it analyzes the results obtained from the data collection step and comparison between SDSC and SDGSC.

4.1 IBSD

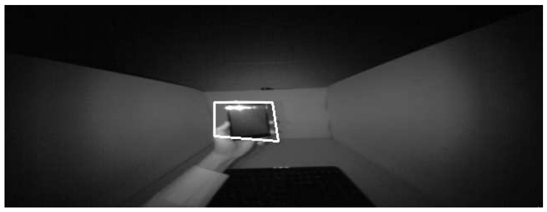
By using the smartphone speckle and traditional image processing algorithms, we were successfully able to quickly localize and recognize a smartphone in a given leap motion image. Although there were certain instances where edges from surrounding objects were incorrectly identified as smartphone edges, IBSD was able to successfully detect a smartphone. There are certain situations where IBSD typically fails, one such



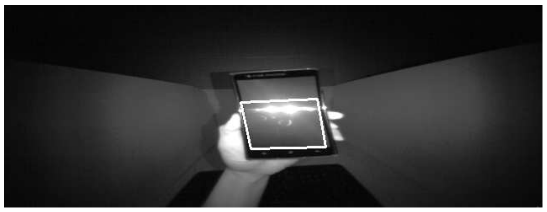
(a)



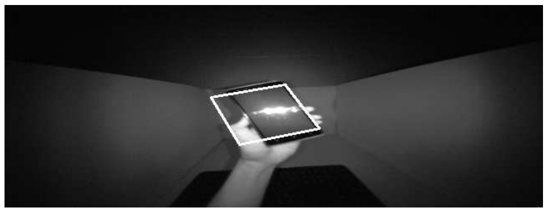
(b)



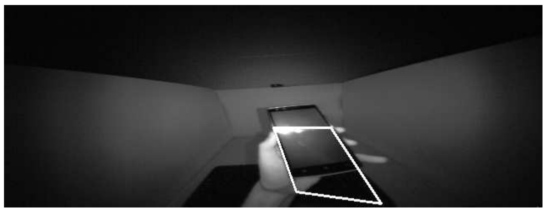
(c)



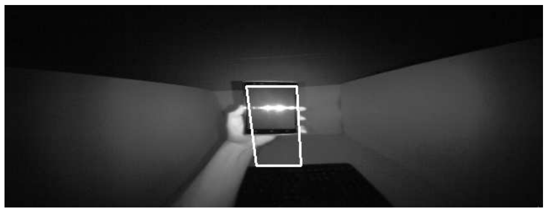
(d)



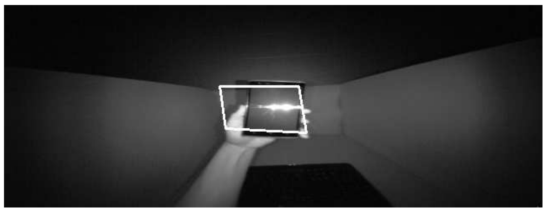
(e)



(f)



(g)



(h)

Figure 4.1: Some sample instances that represent incorrect smartphone detection by IBSD. Note that in these images the edges of other objects in the surroundings are considered part of the smartphone.

instance is when the smartphone's screen is facing the HMD in such an angle that the speckle will not be present on the smartphone. However, this would imply that users are holding the smartphone in such a way that they will not usually be able to read clearly the text displayed on the smartphone's screen. Another instance where IBSD fails is when a similar object with a highly reflective surface is present in the vicinity of the smartphone. In these cases, there will be two speckles present in the leap motion image, and IBSD may produce a false detection when the wrong speckle is chosen. Sample instances of incorrect smartphone detection by IBSD are shown in Figure 4.1.

4.2 SDSC

4.2.1 Feature selection

Different set of features were selected by the two feature selection methods used (RRF and RFE). Based on the correlation between features (see Figure 4.2 and Table 4.1) RRF removed SpeckleX, TopRightX, TopRightY, BottomRightX and BottomRightY, and thus RRF's set of features consisted of six attributes. RFE considered features SpeckleX, SpeckleSize, TopLeftX and TopRightX irrelevant for the construction of the classification model, and thus RFE's set of features consisted of seven attributes. Four features (SpeckleY, TopLeftY, BottomLeftX, and BottomLeftY) were selected by both approaches.

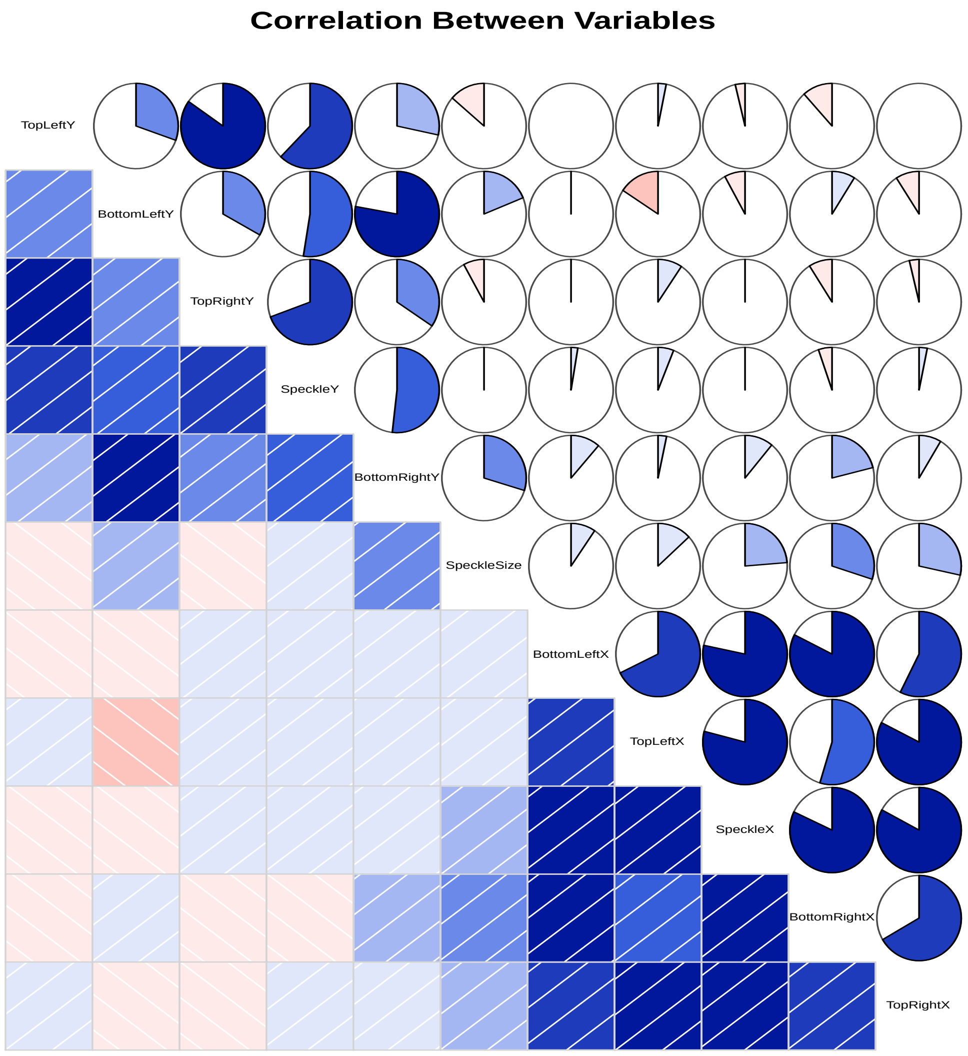


Figure 4.2: Graphical representation of correlation matrix. Darker blue indicates higher correlation, darker red indicates lower correlation, and white indicates no correlation.

Table 4.1: Correlation matrix of all features in SDSC. Values more than ± 0.70 indicates a high correlation.

	SpeckleX	SpeckleY	SpeckleSize	TopLeftX	TopLeftY	TopRightX	TopRightY	BottomRightX	BottomRightY	BottomLeftX	BottomLeftY
SpeckleX	1	0.01	0.237	0.791	-0.038	0.829	0.021	0.82	0.109	0.783	-0.079
SpeckleY	0.01	1	0.021	0.06	0.622	0.031	0.693	-0.052	0.518	0.026	0.525
SpeckleSize	0.237	0.021	1	0.131	-0.136	0.284	-0.08	0.3	0.297	0.095	0.188
TopLeftX	0.791	0.06	0.131	1	0.031	0.825	0.093	0.547	0.033	0.677	-0.157
TopLeftY	-0.038	0.622	-0.136	0.031	1	0.005	0.848	-0.115	0.284	-0.007	0.304
TopRightX	0.829	0.031	0.284	0.825	0.005	1	-0.037	0.666	0.085	0.573	-0.09
TopRightY	0.021	0.693	-0.08	0.093	0.848	-0.037	1	-0.09	0.345	0.023	0.332
BottomRightX	0.82	-0.052	0.3	0.547	-0.115	0.666	-0.09	1	0.21	0.825	0.089
BottomRightY	0.109	0.518	0.297	0.033	0.284	0.085	0.345	0.21	1	0.113	0.779
BottomLeftX	0.783	0.026	0.095	0.677	-0.007	0.573	0.023	0.825	0.113	1	-0.022
BottomLeftY	-0.079	0.525	0.188	-0.157	0.304	-0.09	0.332	0.089	0.779	-0.022	1

Table 4.2: Area under the curve (± SD) for each feature selection method

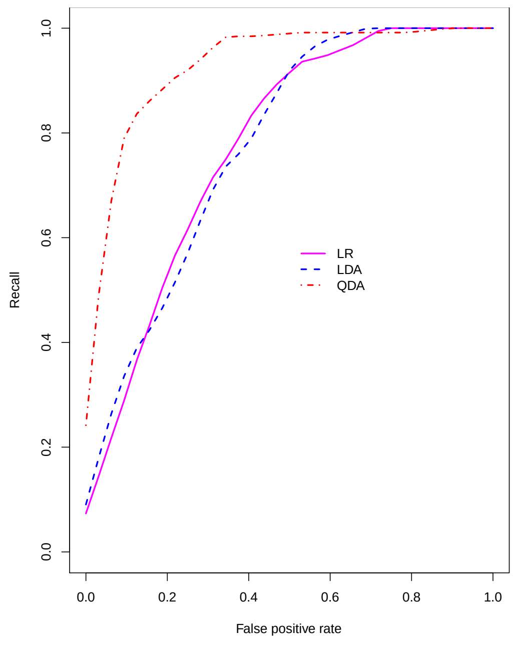
Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.726 ± 0.12	0.727 ± 0.12	0.838 ± 0.07
Recursive Feature Elimination	0.767 ± 0.12	0.771 ± 0.12	0.844 ± 0.08
All available features	0.778 ± 0.13	0.777 ± 0.14	0.935 ± 0.04

Table 4.3: Accuracy (± SD) performance measure for each feature selection method

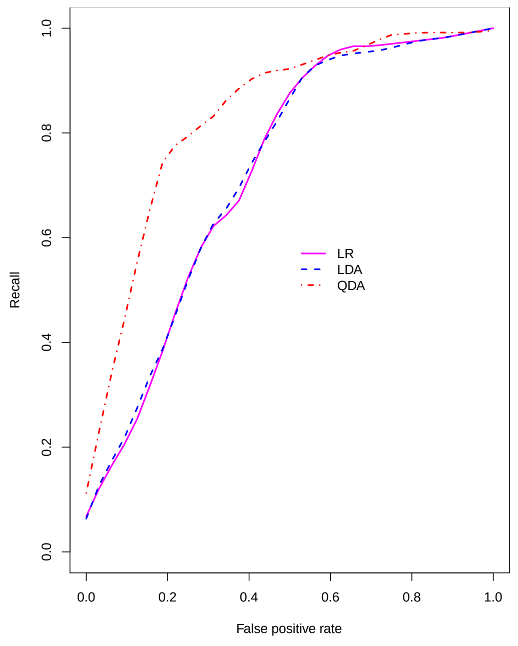
Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.690 ± 0.10	0.686 ± 0.10	0.840 ± 0.05
Recursive Feature Elimination	0.738 ± 0.12	0.732 ± 0.11	0.831 ± 0.06
All available features	0.758 ± 0.13	0.754 ± 0.12	0.907 ± 0.06

4.2.2 Classifier Selection

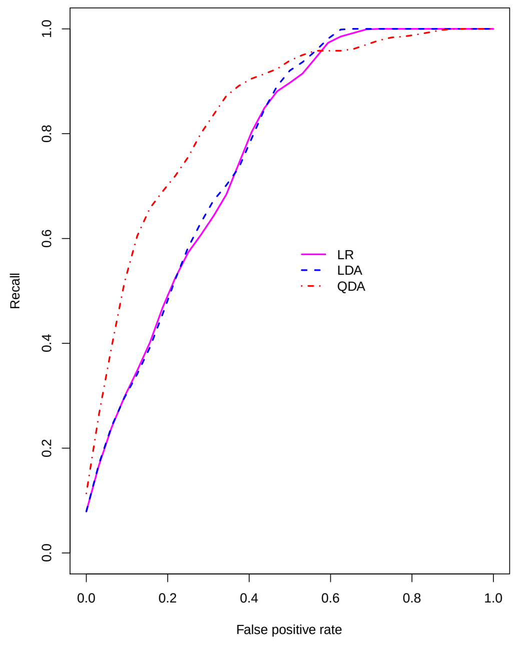
Using all 11 features, we evaluated the performance of LR, LDA and QDA using 10-fold cross-validation. Table 4.6 gives the performance measures averaged out over the ten folds for each classifier. Figure 4.4(a) depicts the ROC curve for each of the classifier. QDA has approximately 16% higher AUC, and higher or comparable recall at the same level of false positive rate (1 - specificity) than LR and LDA. QDA has also the lowest standard deviation for each of the performance measures, which is



(a)



(b)



(c)

Figure 4.3: Average ROC curve for a) Remove Redundant Feature (RRF) selection method b) Recursive Feature Elimination (RFE) method c) all available features used

Table 4.4: Sensitivity (± SD) performance measure for each feature selection method

Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.852 ± 0.12	0.870 ± 0.14	0.854 ± 0.09
Recursive Feature Elimination	0.848 ± 0.09	0.860 ± 0.09	0.864 ± 0.11
All available features	0.875 ± 0.12	0.873 ± 0.09	0.906 ± 0.09

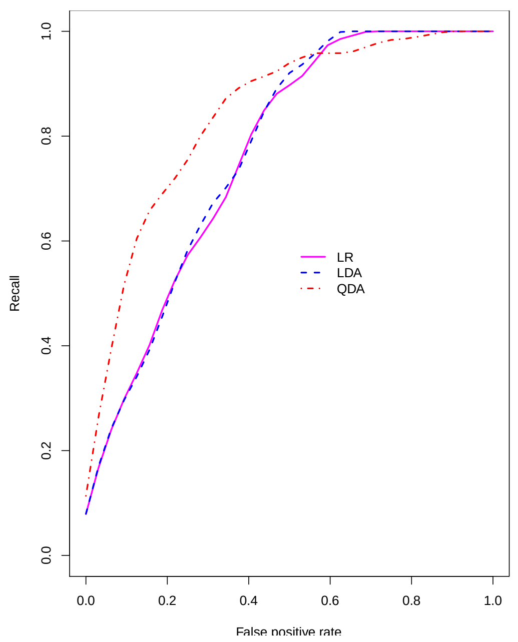
Table 4.5: Specificity (± SD) performance measure for each feature selection method

Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.635 ± 0.13	0.627 ± 0.13	0.830 ± 0.08
Recursive Feature Elimination	0.690 ± 0.14	0.686 ± 0.15	0.816 ± 0.10
All available features	0.710 ± 0.17	0.701 ± 0.17	0.899 ± 0.07

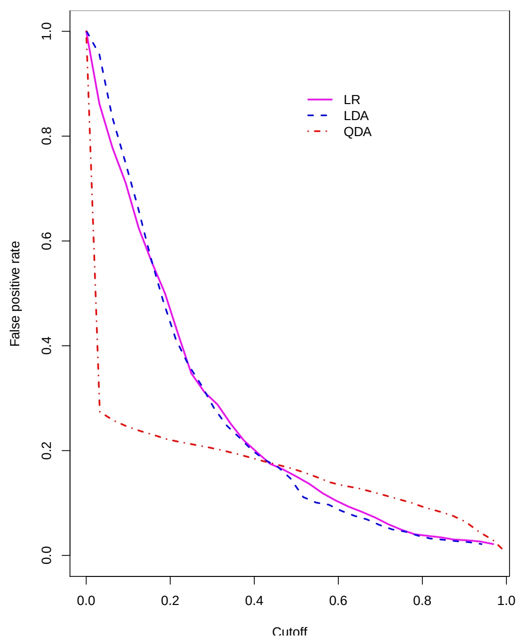
a strong indication of a more robust classifier for this application domain than LR and LDA. Based on these performance measures, we decided to use QDA to classify whether the detected smartphone edges belonged to the smartphone or not.

4.2.3 Comparison of IBSD against SDSC

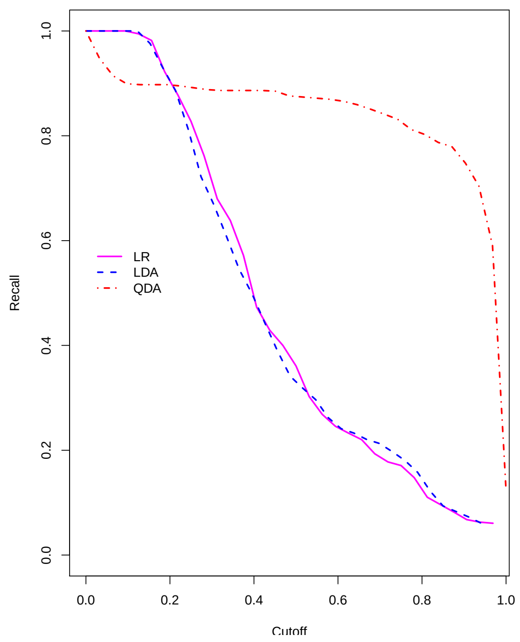
After integrating the mathematical model of the statistical classifier into IBSD, we further compared the performance of IBSD against that of SDSC with a validation data set that was not used during training. To do that, a dataset of 210 records was generated (as discussed in Subsection 3.3.4). Out of the 210 instances, there were 59 (or 28.1%) instances where IBSD considered edges of surrounding objects to



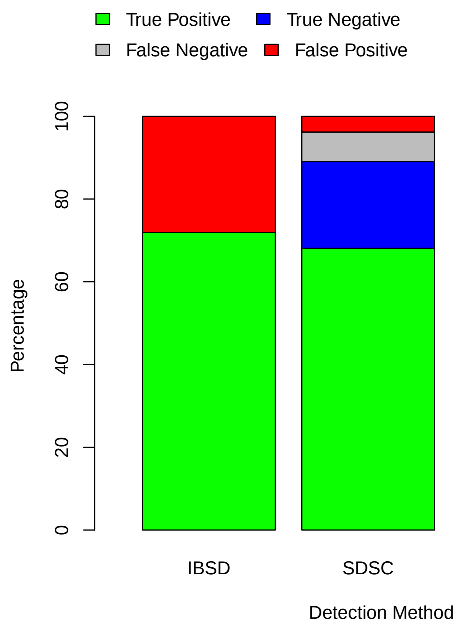
(a)



(b)



(c)



(d)

Figure 4.4: (a) Comparison of LR, LDA and QDA based on ROC curve across various probability thresholds (b) Comparison based on recall (sensitivity)(c) Comparison based on false positive rate (1 -specificity) (d) Comparison between IBSD and SDSC.

Table 4.6: Average performance measures AUC (± SD), Sensitivity (± SD), Specificity (± SD) and Accuracy (± SD) of LR, LDA and QDA over 10 folds. Sensitivity, Specificity and Accuracy of LR, LDA and QDA for each of the 10 folds were calculated for the optimal probability cutoff. The average optimal cutoff for LR, LDA and QDA was 0.295 ± 0.10, 0.280 ± 0.10 and 0.642 ± 0.30 respectively

Performance Measure	LR	LDA	QDA
AUC	0.778 ± 0.13	0.777 ± 0.14	0.935 ± 0.04
Sensitivity	0.875 ± 0.12	0.873 ± 0.09	0.906 ± 0.09
Specificity	0.710 ± 0.17	0.701 ± 0.17	0.899 ± 0.07
Accuracy	0.758 ± 0.13	0.754 ± 0.12	0.907 ± 0.06

be part of the smartphone, while only 8 (or 3.8%) false positives were reported by SDSC. SDSC‘s total error rate was 11%. In the Figure 4.4(d), the y-axis indicates the percentage of instances in the validation set. IBSD error rate was 28.1% (red block) while SDSC error rate was 11% (gray and red blocks).

Comparing IBSD and SDSC, we found that SDSC rejects 75% of the cases that were falsely considered to be smartphone by IBSD. An accuracy of 89% was achieved by SDSC compared to an accuracy of 72% obtained by IBSD. This represents a 17% gain in accuracy and demonstrates the benefits of combining the image-based smartphone detector with the model constructed by QDA. Some instances from the validation dataset where the IBSD incorrectly detects the edges of a smartphone are shown in Figure 4.1. These images show that IBSD considered edges of surrounding objects like keyboard or the cubicle to be smartphone edges. Such images were rejected by SDSC.

4.3 SDGSC

The data collected during the data collection study to train SDSC to accommodate for the different size of smartphones as well as different position in which a user can hold a smartphone, was analyzed by performing feature selection step (as described in 3.3.3.2) and classifier selection step (as described in 3.3.3.3) to check if QDA was still the best performing classifier or not. The results obtained after evaluating the data for best feature selection method and best classifier are given below:

4.3.1 Feature selection

Different set of features were selected by the two feature selection methods used (RRF and RFE). Based on the correlation between features (see Figure 4.5 and Table 4.7) RRF removed SpeckleX, SpeckleY, TopLeftY, and BottomRightY. Note that except for SpeckleX all other removed features were different from the previous results obtained when limited dataset was used. After removal, RRF's set of features consisted of seven attributes. RFE considered all features as relevant for the construction of the classification model which was different from the previous results obtained.

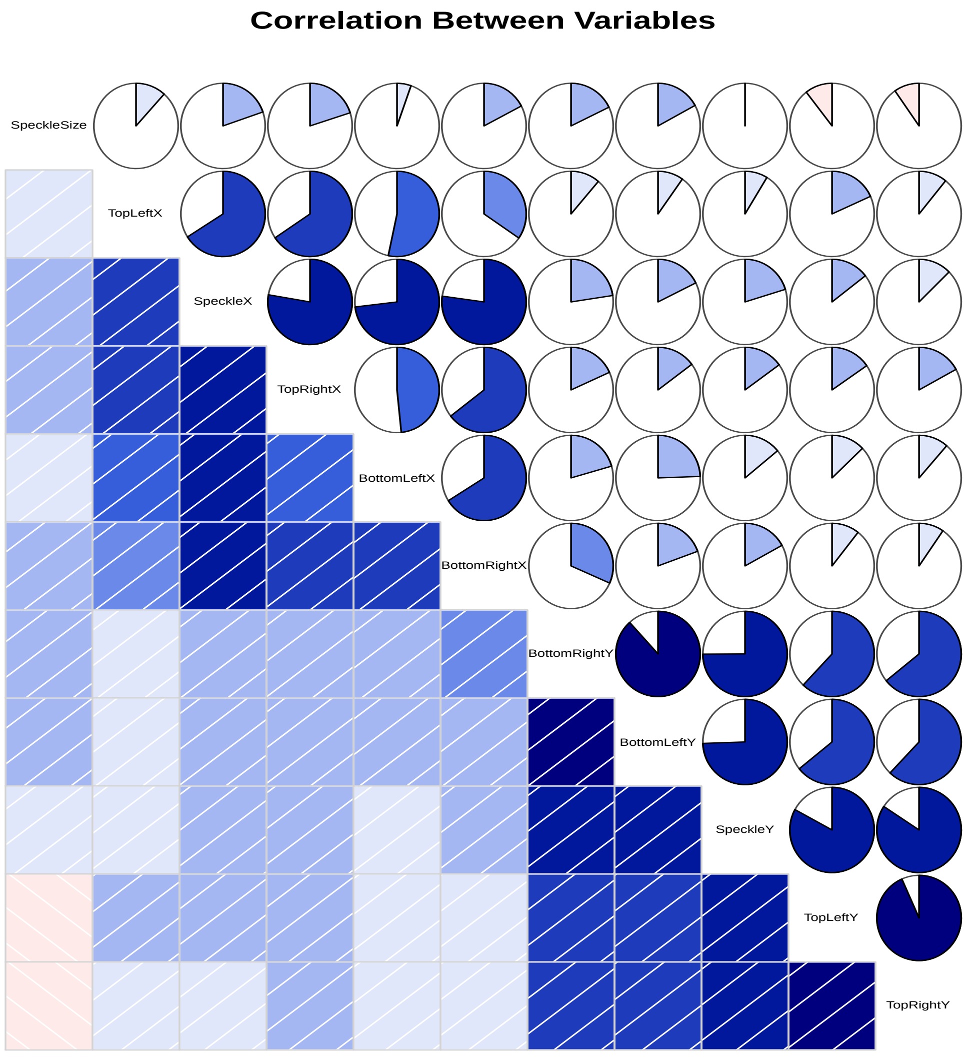


Figure 4.5: Graphical representation of correlation matrix for the Dataset collected from 10 users. Darker blue indicates higher correlation, darker red indicates lower correlation, and white indicates no correlation.

Table 4.7: Correlation matrix of all features in SDGSC for the Dataset collected from 10 users. Values more than ± 0.70 indicate high correlation.

	SpeckleX	SpeckleY	SpeckleSize	TopLeftX	TopLeftY	TopRightX	TopRightY	BottomRightX	BottomRightY	BottomLeftX	BottomLeftY
SpeckleX	1	0.202	0.197	0.659	0.145	0.777	0.126	0.772	0.226	0.731	0.176
SpeckleY	0.202	1	0.021	0.087	0.829	0.15	0.843	0.17	0.749	0.14	0.745
SpeckleSize	0.197	0.021	1	0.116	-0.105	0.2	-0.097	0.173	0.178	0.054	0.17
TopLeftX	0.659	0.087	0.116	1	0.182	0.656	0.108	0.346	0.112	0.533	0.098
TopLeftY	0.145	0.829	-0.105	0.182	1	0.154	0.933	0.106	0.62	0.128	0.642
TopRightX	0.777	0.15	0.2	0.656	0.154	1	0.17	0.645	0.182	0.484	0.148
TopRightY	0.126	0.843	-0.097	0.108	0.933	0.17	1	0.096	0.642	0.114	0.621
BottomRightX	0.772	0.17	0.173	0.346	0.106	0.645	0.096	1	0.316	0.661	0.194
BottomRightY	0.226	0.749	0.178	0.112	0.62	0.182	0.642	0.316	1	0.205	0.882
BottomLeftX	0.731	0.14	0.054	0.533	0.128	0.484	0.114	0.661	0.205	1	0.244
BottomLeftY	0.176	0.745	0.17	0.098	0.642	0.148	0.621	0.194	0.882	0.244	1

Similar to results obtained for SDSC, the AUC was the highest when all available features were considered for all classifiers (LR, LDA and QDA). The AUC performance measure obtained by each classifier for each subset of features is given in Table 4.8. An interesting thing to note is that it seems selection of features does not matter for LR and LDA as AUC is nearly the same for both. Similarly, other performance measures like accuracy (given in Table 4.9), sensitivity (given in Table 4.10) and specificity (given in Table 4.11) were highest when all features were considered. These results prove that selecting all features in training a classifier provides the best possible classifier performance irrespective of smartphone's size or position in which a user can hold a smartphone. The ROC curve for each features subset selection method is given in Figure 4.6.

Table 4.8: Area under the curve (± SD) for each feature selection method

Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.802 ± 0.01	0.757 ± 0.01	0.866 ± 0.01
All available features	0.801 ± 0.01	0.758 ± 0.01	0.892 ± 0.01

Table 4.9: Accuracy (± SD) performance measure for each feature selection method

Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.736 ± 0.02	0.688 ± 0.03	0.787 ± 0.02
All available features	0.736 ± 0.02	0.684 ± 0.03	0.814 ± 0.01

Table 4.10: Sensitivity (± SD) performance measure for each feature selection method

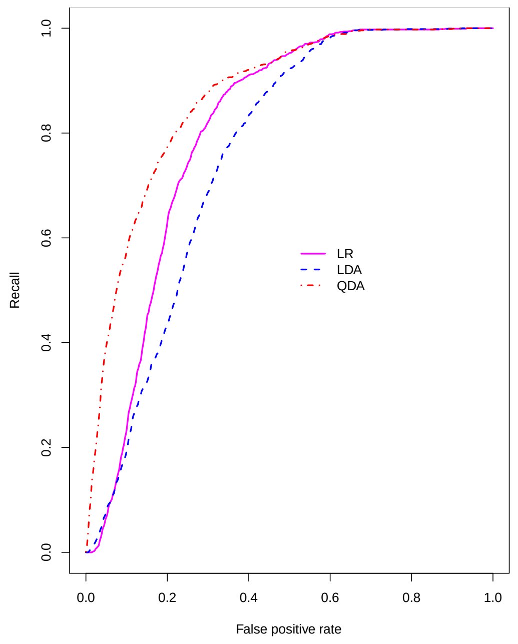
Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.816 ± 0.04	0.764 ± 0.04	0.813 ± 0.03
All available features	0.811 ± 0.03	0.782 ± 0.04	0.837 ± 0.02

Table 4.11: Specificity (± SD) performance measure for each feature selection method

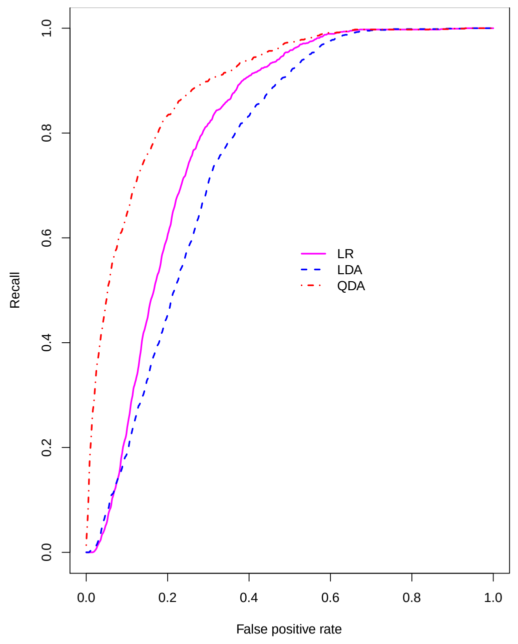
Feature Selection Method	LR	LDA	QDA
Remove Redundant Feature	0.722 ± 0.03	0.675 ± 0.03	0.782 ± 0.02
All available features	0.722 ± 0.02	0.667 ± 0.03	0.809 ± 0.02

4.3.2 Classifier Selection

Using all 11 features, we again evaluated the performance of LR, LDA and QDA using 10-fold cross-validation. Table 4.12 gives the performance measures averaged out over the ten folds for each classifier. Figure 4.7(a) depicts the ROC curve for each of the classifier. In this case, QDA has higher AUC, and higher or comparable recall at the same level of false positive rate (1 - specificity) than LR and LDA (4.7(b)). The result indicates that QDA remains the best classifier even after including dataset from 10 new users.



(a)

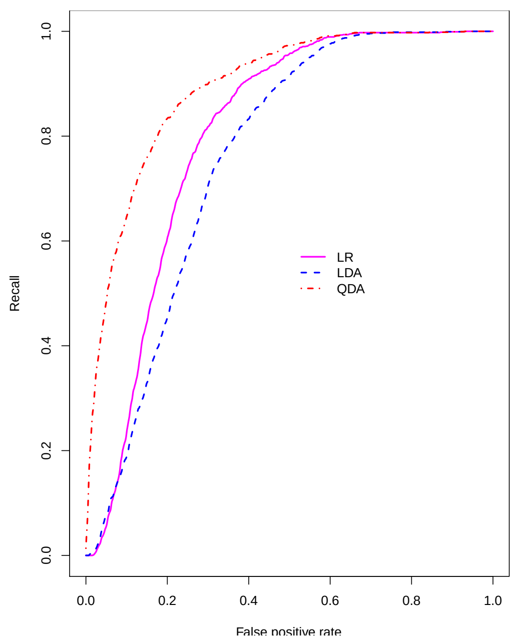


(b)

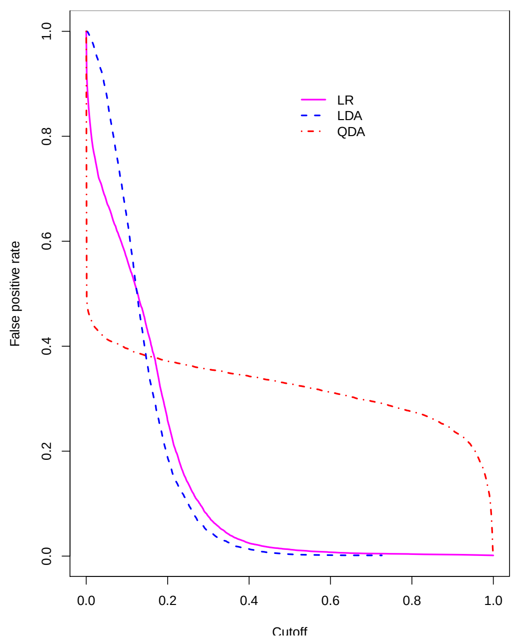
Figure 4.6: Average ROC curve for a) Remove Redundant Features (RRF) selection method b) all available features used

4.3.3 Comparison of SDSC against SDGSC

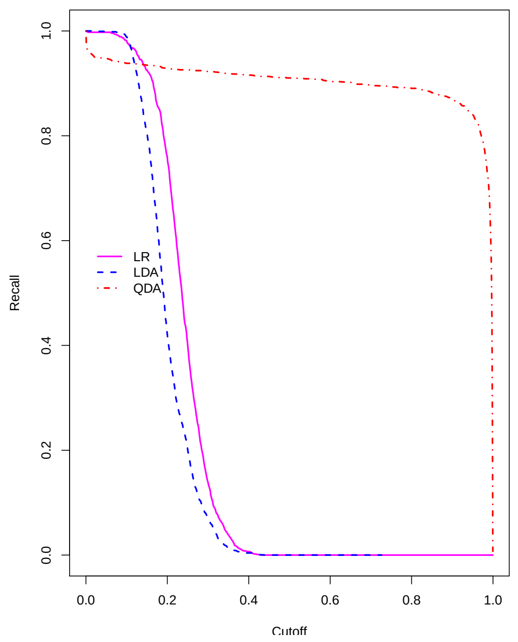
After integrating the new generalized smartphone detector (SDGSC), we compared the performance of the previously obtained SDSC with that of SDGSC with a validation data set that was captured from a new user who was not a participant of the data collection study. A dataset of 1,200 records was recorded. Out of the 1,200 instances, there were 126 (or 10.5%) instances where SDSC considered incorrect edges to be part of smartphone. In comparison, SDGSC considered incorrect edges only in two such instances. Figure 4.7(d) shows the comparison made between SDSC and SDGSC. SDSC's total error rate was 14.7%, while for SDGSC it was 8.8%. Com-



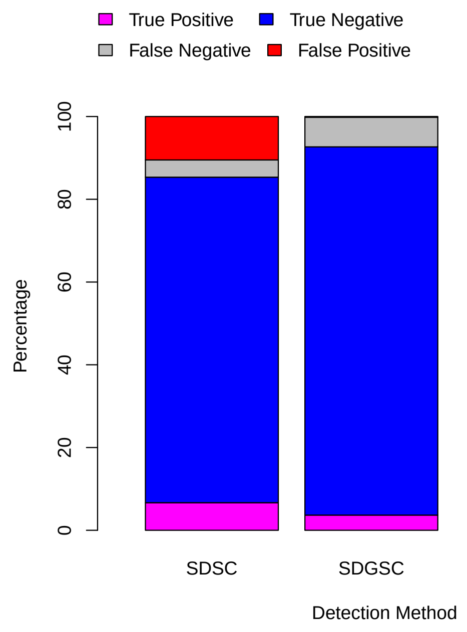
(a)



(b)



(c)



(d)

Figure 4.7: (a) Comparison of LR, LDA and QDA based on ROC curve across various probability thresholds (b) Comparison based on recall (sensitivity)(c) Comparison based on false positive rate (1 -specificity) (d) Comparison between SDSC and SDGSC.

Table 4.12: Average performance measures AUC (± SD), Sensitivity (± SD), Specificity (± SD) and Accuracy (± SD) of LR, LDA and QDA over 10 folds. Sensitivity, Specificity and Accuracy of LR, LDA and QDA for each of the 10 folds were calculated for the optimal probability cutoff. The average optimal cutoff for LR, LDA and QDA was 0.194 ± 0.01, 0.157 ± 0.0 and 0.959 ± 0.01 respectively

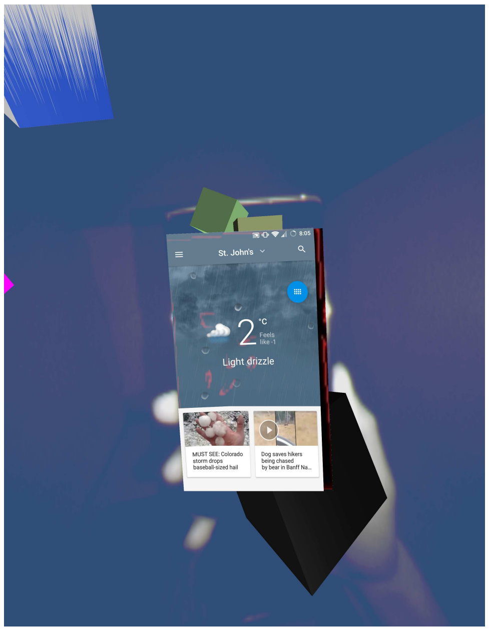
Performance Measure	LR	LDA	QDA
AUC	0.801 ± 0.01	0.758 ± 0.01	0.892 ± 0.01
Sensitivity	0.811 ± 0.03	0.782 ± 0.04	0.837 ± 0.02
Specificity	0.722 ± 0.02	0.667 ± 0.03	0.809 ± 0.02
Accuracy	0.736 ± 0.02	0.684 ± 0.03	0.814 ± 0.01

paring SDSC and SDGSC, we found that SDGSC rejects 98% of the cases that were falsely considered to be smartphone by SDSC. Another point worth noticing is that while SDGSC nearly eliminates all false positives (wrong edges considered as smartphone), the false negative rate increased. That means the new classifier model is more strict in accepting edges to be a part of smartphone. The reduction in the proportion of false positives increases the stability of the designed VR system when displaying the virtual smartphone in the correct position. Our evaluation of the system, after integrating with SDGSC, gave us reduced jumpiness effect which facilitated use of smartphone for typing activities.

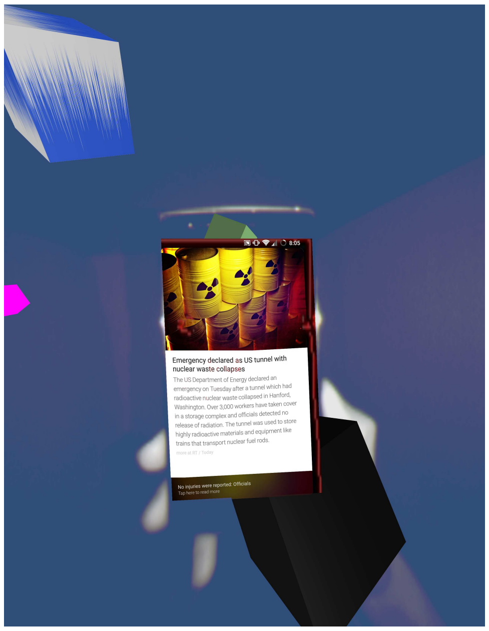
Similar to SDSC, SDGSC also works as an additional filter which discards false smartphone edges detected by IBSD. Therefore, it will still be limited by the inability to detect a smartphone when the speckle is missing. On the other hand, it will perform much better than SDSC at discarding false positives when the wrong speckle is identified.

4.4 Smartphone detection after SDGSC

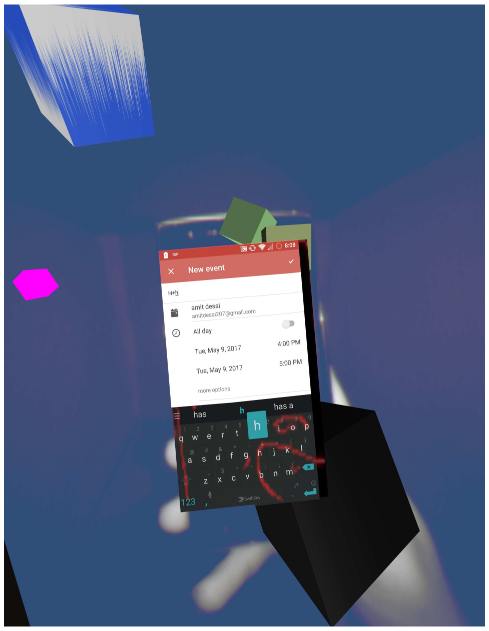
By combining traditional image processing techniques with statistical classifiers, we designed a system that can successfully detect the presence of the smartphone and reject false positive cases 98% of the time. Our empirical assessment indicates that the system is more stable and the virtual smartphone aligns properly with the smartphone present in the leap motion image. The system is also able to map the images obtained from the smartphone's screen to the smartphone displayed inside the VR space, and our initial testing showed that user can operate their smartphone device without having to remove their HMD device. Some instances where the user was able to launch an App, able to read text displayed and attempt to type a message on the smartphone's screen in VR environment are shown in Figure 4.8. These images show that, although limitations exist, the method proposed can bring modern communication devices like smartphones inside the VR space. It also shows that users can operate their smartphones from inside the VR space and no longer have to remove their HMD's each and every time they need to check their social networks or talk with friends and family over the phone.



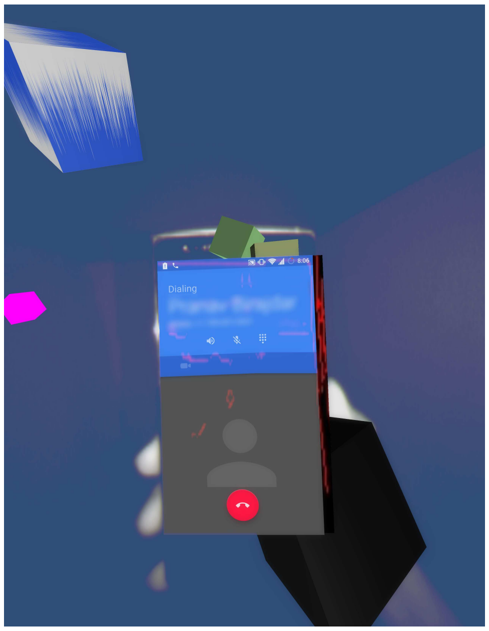
(a)


Emergency declared as US tunnel with nuclear waste collapses

(b)



(c)



(d)

Figure 4.8: Some instances where users were able to interact with the smartphone inside VR environment. (a) User can check weather updates (b) Read news in 'InShorts' app (c) Create an appointment (d) Initiate a phone call

Chapter 5

Conclusions and Future Work

The visual isolation from the real world while wearing an HMD makes the task of using a smartphone difficult. This may in turn lead to temporary isolation, where users are not able to get in touch with their friends and family or check out important notifications when immersed in the VR space. To address this issue, we present a novel method that allows people wearing VR HMDs to use their smartphones without removing their HMDs. The devised method combines traditional image processing techniques with a statistical classifier to accurately detect the presence of the smartphone in the user's real-world surrounding. An additional camera such as the Leap Motion device is used to capture the user's real-world immediate surroundings. Once the smartphone is detected, we augment the scene inside the VR HMD with a view of the user's device so that the user can interact with the device without removing the headset. This setup allows the users to interact with the smartphone or tablet device from within the virtual space and be in touch with their friends and family through it. Although the proposed system has some limitations and the human visual system is quite sensitive to these, achieving a satisfying solution is within reach, and the presented results are encouraging. The proposed system can be further explored for use in application areas where the use of traditional input devices can be challenging and smartphones may come in handy. Through user studies, activities that rely on the smartphones touch screen as an input device, such as text entry, game control, internet browsing, and drawing gestures can be further explored.

5.1 Statistical classifiers in Image Processing

We assessed the accuracy of three statistical classifiers (LR, LDA and QDA) and three feature selection approaches (RRF, RFE, and all features available) for detecting a smartphone in real-time from an image captured by the Leap Motion device. From the three statistical classifiers, QDA showed approximately 13% higher AUC, and higher recall at the same level of false positive rate (< 0.2) than LR and LDA for the initial dataset used. QDA had also the lowest standard deviation for each of the performance measurements, a strong indication of a more robust classifier for this application domain than LR and LDA. Based on these performance measures, we used QDA to classify whether the detected smartphone edges belonged to the smartphone or not at run-time.

From the feature selection approaches, QDA's AUC was statistically significantly higher when all available features were considered (p-value ≤ 0.001, Mann-Whitney test) than when using the features subsets selected by the other feature selection approaches. Based on the results obtained, we decided to use all available features to construct the run-time classification model of the smartphone detector with generalized statistical classifiers (SDGSC). The original smartphone detector (IBSD) had a high false positive rate, incorrectly identifying edges of surrounding objects as smartphone edges. When applying the QDA classifier trained with all available features on the images produced by the image-based smartphone detector, we observed that SDGSC rejects almost 98% of the false positive cases. This demonstrates that statistical classifiers can be effectively used to build a reliable system. In our case we can reliably overlay the smartphone screenshot to the real-world smartphone displayed in VR environment. A reliable smartphone display enables the users to perform different operations like reading text, answering calls or checking the weather from their smartphone from within the VR environment.

5.2 Publication of this Research

The research described in this thesis has been published in the following conference:

- A Window to your Smartphone: Exploring Interaction and Communication in Immersive VR with Augmented Virtuality

  Amit P. Desai, Lourdes Peña-Castillo and Oscar Meruvia-Pastor

Proceeding of the 14th Conference on Computer and Robot Vision (CRV), May 17-19, 2017, Edmonton, Alberta, Canada.

Bibliography

[1] D. Kushner, “Virtual reality’s moment,” *IEEE Spectrum*, vol. 51, no. 1, pp. 34–37, January 2014.

[2] Oculus, “Oculus Rift,” https://www.oculus.com/en-us/rift/, 2016, accessed: 2016-05-01.

[3] HTC, “HTC Vive,” http://www.htcvive.com/, 2016, accessed: 2016-05-01.

[4] Samsung Electronics Co. Ltd., “Samsung Gear VR,” http://www.samsung.com/global/galaxy/gear-vr/, 2016, accessed: 2017-01-21.

[5] Sony Interactive Entertainment, “PlayStation VR,” https://www.playstation.com/en-us/explore/playstation-vr/, 2016, accessed: 2016-05-01.

[6] Google VR, “Google Cardboard,” https://vr.google.com/cardboard/index.html, 2016, accessed: 2016-05-01.

[7] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, S. Davidoff, J. S. Norris, and G. Longo,

"Immersive and collaborative data visualization using virtual reality platforms," in *Big Data (Big Data), 2014 IEEE International Conference on*, Oct 2014, pp. 609–614.

[8] J. A. Henry and N. F. Polys, "The effects of immersion and navigation on the acquisition of spatial knowledge of abstract data networks," *Procedia Computer Science*, vol. 1, no. 1, pp. 1737 – 1746, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050910001961

[9] E. Champion, I. Bishop, and B. Dave, "The palenque project: evaluating interaction in an online virtual archaeology site," *Virtual Reality*, vol. 16, no. 2, pp. 121–139, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10055-011-0191-0

[10] H. Rua and P. Alvito, "Living the past: 3d models, virtual reality and game engines as tools for supporting archaeology and the reconstruction of cultural heritage the case-study of the roman villa of casal de freiria," *Journal of Archaeological Science*, vol. 38, no. 12, pp. 3296 – 3308, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305440311002494

[11] A. S. Mathur, "Low cost virtual reality for medical training," in *2015 IEEE Virtual Reality (VR)*, March 2015, pp. 345–346.

[12] J. P. Bliss, P. D. Tidwell, and M. A. Guest, "The effectiveness of virtual reality for administering spatial navigation training to firefighters," *Presence: Teleoperators*

and Virtual Environments, vol. 6, no. 1, pp. 73–86, 1997.

[13] L. K. Simone, M. T. Schultheis, J. Rebimbas, and S. R. Millis, "Head-mounted displays for clinical virtual reality applications: pitfalls in understanding user behavior while using technology," *Cyberpsychol Behav*, vol. 9, no. 5, pp. 591–602, Oct 2006.

[14] L. Zhang, B. C. Abreu, G. S. Seale, B. Masel, C. H. Christiansen, and K. J. Ottenbacher, "A virtual reality environment for evaluation of a daily living skill in brain injury rehabilitation: reliability and validity1," *Archives of Physical Medicine and Rehabilitation*, vol. 84, no. 8, pp. 1118 – 1124, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000399930300203X

[15] J. Broeren, M. Rydmark, and K. S. Sunnerhagen, "Virtual reality and haptics as a training device for movement rehabilitation after stroke: A single-case study1," *Archives of Physical Medicine and Rehabilitation*, vol. 85, no. 8, pp. 1247 – 1250, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003999303011985

[16] R. W. Lindeman, "A low-cost, low-latency approach to dynamic immersion in occlusive head-mounted displays," in *2016 IEEE Virtual Reality (VR)*, March 2016, pp. 221–222.

[17] M. Billinghurst and H. Kato, "Collaborative mixed reality," in *Proceedings of the International Symposium on Mixed Reality*, Yokohama, Japan, March 1999, pp. 261–284.

[18] M. McGill, D. Boland, R. Murray-Smith, and S. Brewster, "A dose of reality: Overcoming usability challenges in VR head-mounted displays," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI 2015. New York, NY, USA: ACM, 2015, pp. 2143–2152. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702382

[19] P. J. Metzger, "Adding reality to the virtual," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, Sep 1993, pp. 7–13.

[20] Smartinsights, "Social Media Stats," http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/, accessed: 2016-05-01.

[21] Leap Motion, "Leap Motion VR," https://www.leapmotion.com/product/vr, 2016, accessed: 2016-05-01.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[23] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2564–2571. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126544

[25] H. Schneiderman and T. Kanade, “A statistical method for 3d object detection applied to faces and cars,” in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 1, 2000, pp. 746–751 vol.1.

[26] A. Takeuchi, S. Mita, and D. McAllester, “On-road vehicle tracking using deformable object model and particle filter with integrated likelihoods,” in *2010 IEEE Intelligent Vehicles Symposium*, June 2010, pp. 1014–1021.

[27] D. Böhning, “Multinomial logistic regression algorithm,” *Annals of the Institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, Mar 1992. [Online]. Available: https://doi.org/10.1007/BF00048682

[28] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, “Fisher discriminant analysis with kernels,” in *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, Aug 1999, pp. 41–48.

[29] S. Srivastava, M. R. Gupta, and B. A. Frigyik, “Bayesian quadratic discriminant analysis,” *Journal of Machine Learning Research*, vol. 8, no. Jun, pp. 1277–1305, 2007.

[30] P. Milgram and F. Kishino, “A taxonomy of mixed reality visual displays,” *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[31] P. Milgram and H. Colquhoun, “A taxonomy of real and virtual world display integration,” *Mixed reality: Merging real and virtual worlds*, vol. 1, pp. 1–26, 1999.

[32] Microsoft, “Kinect for Xbox One,” http://www.xbox.com/en-CA/xbox-one/accessories/kinect, 2017, accessed: 2017-04-29.

[33] H. Regenbrecht, T. Lum, P. Kohler, C. Ott, M. Wagner, W. Wilke, and E. Mueller, “Using augmented virtuality for remote collaboration,” *Presence: Teleoperators and Virtual Environments - Special issue: Advances in collaborative virtual environments*, vol. 13, no. 3, pp. 338–354, Jul. 2004. [Online]. Available: http://dx.doi.org/10.1162/1054746041422334

[34] F. Steinicke, G. Bruder, K. Rothaus, and K. Hinrichs, “Poster: A virtual body for augmented virtuality by chroma-keying of egocentric videos,” in *3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on*, March 2009, pp. 125–126.

[35] F. Tecchia, G. Avveduto, R. Brondi, M. Carrozzino, M. Bergamasco, and L. Alem, "I'm in vr!: Using your own hands in a fully immersive mr system," in *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '14. New York, NY, USA: ACM, 2014, pp. 73–76. [Online]. Available: http://doi.acm.org/10.1145/2671015.2671123

[36] T. Ha, S. Feiner, and W. Woo, "Wearhand: Head-worn, rgb-d camera-based, bare-hand user interface with visually enhanced depth perception," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Sept 2014, pp. 219–228.

[37] G. Bruder, F. Steinicke, K. Rothaus, and K. Hinrichs, "Enhancing presence in head-mounted display environments by visual body feedback using head-mounted cameras," in *Proceedings of the 2009 International Conference on CyberWorlds*, ser. CW '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 43–50. [Online]. Available: http://dx.doi.org/10.1109/CW.2009.39

[38] T. Gnther, I. S. Franke, and R. Groh, "Aughanded virtuality - the hands in the virtual environment," in *2015 IEEE Virtual Reality (VR)*, March 2015, pp. 327–328.

[39] G. Bruder, F. Steinicke, D. Valkov, and K. Hinrichs, "Immersive virtual studio for architectural exploration," in *2010 IEEE Symposium on 3D User Interfaces (3DUI)*, March 2010, pp. 125–126.

[40] L. P. Fiore and V. Interrante, “Towards achieving robust video selfavatars under flexible environment conditions,” *International Journal of Virtual Reality*, vol. 11, no. 3, pp. 33–41, 2012.

[41] D. Nahon, G. Subileau, and B. Capel, “Never blind vr - enhancing the virtual reality headset experience with augmented virtuality,” in *2015 IEEE Virtual Reality (VR)*, March 2015, pp. 347–348.

[42] C. Choi and H. I. Christensen, “3d textureless object detection and tracking: An edge-based approach,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 3877–3884.

[43] F. Tombari, A. Franchi, and L. Di, “Bold features to detect texture-less objects,” in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 1265–1272.

[44] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, “Dominant orientation templates for real-time detection of texture-less objects,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 2257–2264.

[45] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, “Gradient response maps for real-time detection of textureless objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, May 2012.

[46] C. Steger, "Occlusion, clutter, and illumination invariant object recognition," *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 34, no. 3/A, pp. 345–350, 2002.

[47] M. Ulrich, C. Wiedemann, and C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1902–1914, Oct 2012.

[48] C. F. Olson and D. P. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Transactions on Image Processing*, vol. 6, no. 1, pp. 103–113, Jan 1997.

[49] F. Hausdorff, "Dimension und äußeres maß," *Mathematische Annalen*, vol. 79, no. 1, pp. 157–179, 1918. [Online]. Available: http://dx.doi.org/10.1007/BF01457179

[50] S. Lee, S. Lee, J. Lee, D. Moon, E. Kim, and J. Seo, "Robust recognition and pose estimation of 3d objects based on evidence fusion in a sequence of images," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 3773–3779.

[51] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'77. San Francisco, CA, USA:

Morgan Kaufmann Publishers Inc., 1977, pp. 659–663. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622943.1622971

[52] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3d camera tracking," in *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nov 2004, pp. 48–56.

[53] C. Kemp and T. Drummond, "Dynamic measurement clustering to aid real time tracking," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, Oct 2005, pp. 1500–1507 Vol. 2.

[54] D. Schreiber, C. Beleznai, and M. Rauter, "Gpu-accelerated human detection using fast directional chamfer matching," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2013, pp. 614–621.

[55] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227 – 248, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0146664X80900544

[56] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *In VISAPP International Conference on Computer Vision Theory and Applications*, 2009, pp. 331–340.

[57] J. Chan, J. A. Lee, and Q. Kemao, "Border: An oriented rectangles approach to texture-less object recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2855–2863.

[58] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[59] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[60] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 577–584. [Online]. Available: http://dl.acm.org/citation.cfm?id=2968618.2968690

[61] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[62] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sept 2010.

[63] Z. Zhang, Y. Liu, A. Li, and M. Wang, "A novel method for user-defined human posture recognition using kinect," in *7th International Congress on Image and*

Signal Processing (CISP), 2014. IEEE, Oct 2014, pp. 36–740.

[64] G. Marqués and K. Basterretxea, "Efficient algorithms for accelerometer-based wearable hand gesture recognition systems," in *13th IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2015*, Oct 2015, pp. 132–139.

[65] T. S. Hai, L. H. Thai, and N. T. Thuy, "Facial expression classification using artificial neural network and k-nearest neighbor," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 7, no. 3, p. 27, February 2015.

[66] Michael Lyons, Miyuki Kamachi, and Jiro Gyoba, "The Japanese Female Facial Expression (JAFFE) Database," http://www.kasrl.org/jaffe.html, 2017, accessed: 2017-04-29.

[67] R. Moreira, A. Magalhães, and H. P. Oliveira, *A Kinect-Based System to Assess Lymphedema Impairments in Breast Cancer Patients.* Cham: Springer International Publishing, 2015, pp. 228–236.

[68] D. R. Nayak, R. Dash, and B. Majhi, "Classification of brain mr images using discrete wavelet transform and random forests," in *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, Dec 2015, pp. 1–4.

[69] S. Das, M. Chowdhury, and M. K. Kundu, "Brain mr image classification using multiscale geometric analysis of ripplet," *Progress In Electromagnetics Research*, vol. 137, pp. 1–17, 2013.

[70] D. T. Larose, *k-Nearest Neighbor Algorithm.* John Wiley & Sons, Inc., 2005, pp. 90–106. [Online]. Available: http://dx.doi.org/10.1002/0471687545.ch5

[71] Oculus VR LLC, "Oculus Rift DK2 Recommended Specifications," https://product-guides.oculus.com/en-us/documentation/dk2/latest/, 2017, accessed: 2017-04-29.

[72] Google, "Android Studio - The Official IDE for Android," https://developer.android.com/studio/index.html, 2017, accessed: 2017-04-29.

[73] Unity Technologies, "Unity - Game Engine," https://unity3d.com/, 2017, accessed: 2017-04-29.

[74] Unity Technologies, "Unity download archive," https://unity3d.com/get-unity/download/archive, 2017, accessed: 2017-04-29.

[75] Leap Motion, Inc, "Leap Motion Unity Assets," https://developer.leapmotion.com/releases/?category=unity, 2017, accessed: 2017-04-29.

[76] Unity Technologies, "Primitive and Placeholder Objects," https://docs.unity3d.com/Manual/PrimitiveObjects.html, 2017, accessed: 2017-04-29.

[77] OpenCV, "OpenCV," http://opencv.org/, 2017, accessed: 2017-04-29.

[78] Emgu CV, "Emgu CV - cross platform .Net wrapper to OpenCV ," http://www.emgu.com/wiki/index.php/Main_Page, 2017, accessed: 2017-04-29.

[79] OpenCV Team, "Basic Thresholding Operations in OpenCV ," http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html, 2017, accessed: 2017-04-29.

[80] OpenCV Team, "SimpleBlobDetector Class Reference in OpenCV," http://docs.opencv.org/trunk/d0/d7a/classcv_1_1SimpleBlobDetector.html, 2017, accessed: 2017-04-29.

[81] M. M. Meyers, "Blur filter for eliminating aliasing in electrically sampled images," Oct 1997, uS Patent 5,682,266.

[82] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[83] N. Kiryati, Y. Eldar, and A. Bruckstein, "A probabilistic hough transform," *Pattern Recognition*, vol. 24, no. 4, pp. 303 – 316, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/003132039190073E

[84] L. I. Kuncheva, *Statistical classifiers.* Heidelberg: Physica-Verlag HD, 2000, pp. 37–78. [Online]. Available: http://dx.doi.org/10.1007/978-3-7908-1850-5_3

[85] Amit P. Desai, "Online Repository for Thesis Material," https://amitdesai207@bitbucket.org/amitdesai207/a-window-to-your-smartphone.git, 2017, accessed: 2017-04-29.

[86] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation.* Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-39940-9_565

[87] M. Kuhn, "Building predictive models in R using the caret package," *Journal of Statistical Software*, vol. 28, no. 1, pp. 1–26, 2008. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v028i05

[88] V. Svetnik, A. Liaw, C. Tong, and T. Wang, *Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 334–343. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-25966-4_33

[89] OnePlus, "OnePlus One," https://oneplus.net/ca_en/one, 2017, accessed: 2017-04-29.

[90] LG Electronics, "LG G4," http://www.lg.com/ca_en/cell-phones/g4, 2017, accessed: 2017-04-29.

[91] OnePlus, "OnePlus 2," https://oneplus.net/ca_en/2, 2017, accessed: 2017-04-29.

[92] LG Electronics, “LG G3,” http://www.lg.com/ca_en/cell-phones/lg-LGD852G, 2017, accessed: 2017-04-29.

[93] HTC Corporation, “HTC One M8,” http://www.htc.com/us/smartphones/htc-one-m8/, 2017, accessed: 2017-04-29.

[94] LG Electronics, “Google Nexus5,” http://www.lg.com/us/cell-phones/lg-D820-Sprint-Black-nexus-5, 2017, accessed: 2017-04-29.

[95] Motorola Mobility LLC., “Moto G,” https://www.motorola.ca/products/moto-g-gen-3, 2017, accessed: 2017-04-29.

Appendix A

Consent for Data Collection Study

Informed Consent Form

Title:	*Data collection to explore communication and interaction in immersive VR environments using smartphones*
Researcher(s):	*Amit Desai* *Department of Computer Science* *Memorial University of Newfoundland* apd525@mun.ca
Supervisor(s):	*Dr. Oscar Meruvia-Pastor* *Department of Computer Science* *Office of the Dean of Science* *Memorial University of Newfoundland* *St. John's, Canada* oscar@mun.ca
	Dr. Lourdes Peña-Castillo *Department of Computer Science* *Department of Biology* *Memorial University of Newfoundland* *St. John's, Canada* lourdes@mun.ca

You are invited to take part in a research project entitled *"Data collection to explore communication and interaction in immersive VR environments using smartphones."*

This form is part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. It also describes your right to withdraw from the study. To decide whether you wish to participate in this research study, you should understand enough about its risks and benefits to be able to make an informed decision. This is the informed consent process. Take time to read this carefully and to understand the information given to you. Please contact the researcher, Amit Desai, if you have any questions about the study or would like more information before you consent.

It is entirely up to you to decide whether to take part in this research. If you choose not to take part in this research or if you decide to withdraw from the research once it has started, there will be no negative consequences for you, now or in the future.

Introduction:

I, Amit Desai, am conducting this research as the Principal Investigator, under the supervision of Dr. Oscar Meruvia-Pastor & Dr. Lourdes Peña-Castillo. This research is part of my Master's Program.

Virtual Reality headsets like Oculus Rift have visual and social isolation problems, preventing users from seeing and sometimes hearing their surrounding environment when immersed in the virtual environment. The focus of this research is to enable the users to use their smartphones without leaving the virtual environment.

Purpose of study:

Collection of the data to train the designed VR system to improve its reliability to correctly identify the smartphone and to account for variations in the way users use their smartphones at different positions and orientations.

What you will do in this study:

The users will be asked to perform following steps:

1. Wear Oculus VR Head-mounted display
2. Pick up the smartphone placed on table or handed to you
3. Place the smartphone in 5 different regions (center plus corners, clockwise from the top left) and at 3 different depths/distances for each region designated in the VR space for a test run.
4. Place the smartphone in 5 different regions (center plus corners) and at 3 different depths/distances for each region designated in the VR space for actual data collection.
5. The previous step will be repeated with the phone in two basic orientations analogous to the portrait and landscape printing orientations. At each designated position, data points from the captured smartphone image will be recorded.

Length of time:

The total amount of time that we expect each user to spend on data collection is 15 minutes.

Withdrawal from the study:

Participation is entirely voluntary and participants can withdraw anytime. At the time of withdrawal, if the data gathered upon that point is not complete, it will be purged. However, the data cannot be removed from the study if a participant withdraws after the training of the system is complete.

Possible benefits:

a) For participants: The students participating in the study will have a firsthand exposure to Virtual Reality systems.
b) For the scientific community: This research focuses on addressing some of the isolation problems suffered by users of immersive VR with the help of smartphones.

Possible risks:
Use of Head Mounted Displays (HMDs) for extended period of time may cause some participants to experience motion sickness or simulation sickness. Participants suffering with motion sickness can feel fatigue, headache, discomfort, difficulty in focusing, dizziness or nausea. To ensure that participants do not experience severe motion sickness symptoms, the participant will be asked to answer Simulator Sickness Questionnaire (SSQ) at various points. If Simulator Sickness symptoms persist, participants will be instructed not to drive back home and rest in the lab or remain on campus until the symptoms subside. If the symptoms still persist 30 minutes after completing the study, participants will be instructed to visit the Student Health Service at the university.

Confidentiality:
During this study no information about the identity of participants will be used either during the analysis of the data or the release of the findings. The participants' names or identifying information will be recorded only in the informed consent form and will be kept confidential. The participants will be assigned random identification numbers in the internal computer systems, statistics analysis and in the release of the findings.

Upon completion of the study the informed consent forms will be archived in the office of the Principal Supervisor. These forms will be kept for a minimum of five years and will be destroyed after that.

Anonymity:
Participant's identity will be kept anonymous. Participants will not be mentioned in the thesis or publication.

Storage of Data:
Upon completion of the study, the informed consent forms will be archived in the office of the Principal Supervisor. All data collected in the study will be kept for a minimum of five years, as required by Memorial University's policy on Integrity in Scholarly Research.

Reporting of Results:
The results, observations and conclusions of the study will be included in the thesis and associated publications. Since the data collected from the participants will be anonymous, none of the personal identification details of the participants will be mentioned anywhere in the thesis.

Sharing of Results with Participants:
The results, observations and conclusions of the study will be included in thesis report and will be available for public viewing. After the results are published, an email with a link to the published results will be sent out to only those participants who had provided their emails in the consent form.

Questions:
The participant is welcome to ask questions any time before, during, or after participation in this research. If the participant would like more information about this study, please contact: Amit Desai, Email: apd525@mun.ca or Dr. Oscar Meruvia-Pastor, Email: oscar@mun.ca.

The proposal for this research has been reviewed by the Interdisciplinary Committee on Ethics in Human Research and found to be in compliance with Memorial University's ethics policy. If you have ethical concerns about the research, such as the way you have been treated or your rights as a participant, you may contact the Chairperson of the ICEHR at icehr@mun.ca or by telephone at 709-864-2861.

Consent:
Your signature on this form means that:

- You have read the information about the research.
- You have been able to ask questions about this study.
- You are satisfied with the answers to all your questions.
- You understand what the study is about and what you will be doing.
- You understand that you are free to withdraw participation in the study without having to give a reason, and that doing so will not affect you now or in the future.
- You understand that if you choose to end participation **during** data collection, any data collected from you up to that point will be destroyed.
- You understand that your data is being collected anonymously and therefore cannot be removed once data collection has ended.

By signing this form, you do not give up your legal rights and do not release the researchers from their professional responsibilities.

Your signature confirms:

☐ I have read what this study is about and understood the risks and benefits. I have had adequate time to think about this and had the opportunity to ask questions and my questions have been answered.

☐ I agree to participate in the research project understanding the risks and contributions of my participation, that my participation is voluntary, and that I may end my participation.

☐ A copy of this Informed Consent Form has been given to me for my records.

Email Consent

Would you like to receive a copy of the published result of this research? ☐ Yes ☐ No

If yes, please provide your email address ____

______________________________ ______________________________

Signature of participant Date

Researcher's Signature:

I have explained this study to the best of my ability. I invited questions and gave answers. I believe that the participant fully understands what is involved in being in the study, any potential risks of the study and that he or she has freely chosen to be in the study.

______________________________ ______________________________

Signature of Principal Investigator Date

Appendix B

Recruitment Poster for Data Collection Study

Participants Needed for a VR Study!

- *We need 10 enthusiastic volunteers to collect important data for our research*
- *Total time required to complete: approximately 15 minutes*
- *Live demonstration on how to perform the task will be provided to all participants*
- *Participants can withdraw from experiment at any time*
- *Identifiable info won't be collected from participants*
- ***Contact: Amit Desai (apd525@mun.ca or 709-682-6617)***

Details:

This Master's program research focuses on allowing the use of smartphones while wearing Head Mounted Displays (HMDs) like the Oculus Rift.

Volunteers will help me train the system to make it robust against different smartphones of various sizes, and holding positions.

Supervisor: Dr. Oscar Meruvia-Pastor (oscar@mun.ca)

Dr. Lourdes Peña-Castillo (lourdes@mun.ca)

The proposal for this research has been reviewed by the Interdisciplinary Committee on Ethics in Human Research and found to be in compliance with Memorial University's ethics policy. If you have ethical concerns about the research, such as the way you have been treated or your rights as a participant, you may contact the Chairperson of the ICEHR at icehr.chair@mun.ca or by telephone at 709-864-2861

Appendix C

Exit Questionnaire for Data Collection Study

No______________ Date____________________

SIMULATOR SICKNESS QUESTIONNAIRE

Instructions: Circle how much each symptom below is affecting you right now.

1. General discomfort	None	Slight	Moderate	Severe
2. Fatigue	None	Slight	Moderate	Severe
3. Headache	None	Slight	Moderate	Severe
4. Eye strain	None	Slight	Moderate	Severe
5. Difficulty focusing	None	Slight	Moderate	Severe
6. Salivation increasing	None	Slight	Moderate	Severe
7. Sweating	None	Slight	Moderate	Severe
8. Nausea	None	Slight	Moderate	Severe
9. Difficulty concentrating	None	Slight	Moderate	Severe
10. Fullness of the Head	None	Slight	Moderate	Severe
11. Blurred vision	None	Slight	Moderate	Severe
12. Dizziness with eyes open	None	Slight	Moderate	Severe
13. Dizziness with eyes closed	None	Slight	Moderate	Severe
14. *Vertigo	None	Slight	Moderate	Severe
15. **Stomach awareness	None	Slight	Moderate	Severe
16. Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation with respect to vertical upright.
** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea.

Source:
Kennedy, R.S., Lane, N.E., Berbaum, K.S., & Lilienthal, M.G. (1993). Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. *International Journal of Aviation Psychology, 3*(3), 203-220.