# 3D Visualization of 2D/360° Image and Navigation in Virtual Reality through Motion Processing via Smart Phone Sensors

| | |
|---|---|
| **Maliha Tasnim Aurini** | **14101051** |
| **Shitab Mushfiq-ul Islam** | **14101088** |

**Supervisor: Dr. Md. Ashraful Alam**

**Department of Computer Science and Engineering,**
**BRAC University**

# Declaration

We, hereby declare that this report is based on our own work and research for our under graduation final thesis. The contents of the report are prepared by us for the thesis and sources of information and other materials that we used for help are acknowledged and mentioned here by reference. Thus, this report has not been submitted anywhere for any degree or award before.

**Signature of Supervisor:**

______________________

Dr. Md. Ashraful Alam
Assistant Professor
Department of Computer Science and Engineering
BRAC University

**Signature of Authors:**

______________________

Maliha Tasnim Aurini

______________________

Shitab Mushfiq-ul Islam

# Acknowledgements

Firstly we need to express gratitude toward Almighty ALLAH to begin with, the maker and the proprietor of this universe, who gave us direction, quality and capacity to finish this thesis. We also would like to thank our thesis advisor Dr. Md. Ashraful Alam for his valuable guidance, feedback and support till now. We are indebted to our parents, friends and teachers for their immense support and for the help they offered during various phase of our thesis.

Maliha Tasnim Aurini, Shitab Mushfiq-ul Islam

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| VR | Virtual Reality |
| 2D | Two Dimensional |
| 3D | Three Dimensional |
| SDK | Software Development Kit |
| IDE | Integrated Development Environment |
| IR | Infrared |

# Abstract

The 360° images or regular 2D images look appealing in Virtual Reality yet they fail to represent depth and how the depth can be used to give an experience to the user from two dimensional images. We proposed an approach for creating stereogram from computer generated depth map using approximation algorithm and later use these stereo pairs for giving a complete experience on VR along with forward and backward navigation using mobile sensors. Firstly the image is being segmented into two images from which we generated our disparity map and afterwards generate the depth image from it. After the creation of the depth image, stereo pair which is the left and right image for the eyes were created. Acquired image from the previous process then handled by Cardboard SDK for VR support used in the Android devices using Google Cardboard headset. With the VR image in the stereoscopic device, we use the accelerometer sensor of the device to determine the movement of the device while head mounted. Unlike the other VR navigation systems offered (HTC Vibe, Oculus) using external sensors, our approach is to use the built-in sensors for motion processing. Using the accelerometer reading from the movement, the user will be able to move around virtually in the constructed image. The results of this experiment are the visual changes of the image displayed in VR according to the viewer's physical movement.

**Keywords:** Virtual reality, stereogram, depth map, accelerometer sensor, navigation, motion processing.

Chapter 01

# Introduction

## 1.1 Motivations and Goals

The purpose of our model is to make a better Virtual Reality Experience bringing new elements that has not been introduced in mobile virtual reality [1]. Generation 3D virtual experience from a single 2D image through a computer generated system is highly unlikely, due to the lack of depth information and dimensional limitations [2]. Therefore it is something we can work on to fill the gaps using estimated data from the limited information we have. Alike decoders where more data can be retrieved from each bits, here we tried to do the same with a 2D image, processing it through various methods to get/estimate the extra depth information to make it a 3D visual.

Our proposed model is based on Virtual Reality of smartphone devices. Since smartphones are accessible to all, it is a platform for Virtual Reality with many unexplored grounds. Navigation in a 3D space using mobile device is still under experiments, since it is only possible using external sensors. The existing technologies use external infrared sensors to map movement and navigation of the head-mount display whereas the most commonly used device, mobile phones are less in use for navigation in VR.

Our approach will allow us to create more possibilities to improve virtual reality experience along with precise navigation using internal sensors. The challenge here is to make sure the motion of the device is processed correctly and the navigation within the image runs smoothly and efficiently.

## 1.2 Scopes

Our approach basically is to make a 2D image feel like a 3D space. Virtual Reality mainly targeted on that from the beginning yet a lot of work has to be done in order to get the optimum experience Virtual Reality promises to do. 3D elements can be easily visualized, accessed and manipulated in VR but a 2D image is still in visual and not accessed. Our experiments unlocks those aspect that a 2D image can be accessed in such way as a 3D image without the cost of 3D visual requirements. Navigating in a 2D image as if it is 3D can open up VR to various applications like Crime Scene Investigations, Street navigation, Simulation and interactive movies etc. It can also lessen cost in

Google Street View, where Street View VR uses two images to show two points in a place, in our approach only one image can give the same or close to same experience.

## 1.3 Overview

Virtual Reality has become a fairly new subfield of visual imagery. Our area of work offers image processing for head-mounted displays along with the idea of having internal motion sensors for depth navigation. Previously Virtual Reality worked with the 360° surround view, and now we are intending to establish the ground where not only to work with the surround view, but also to go in depth and navigate to a certain extend of depth using extra information extracted from the image. It is important we explore the possibilities of Virtual Reality for a better experience of view or simulation that can have multiple real life applications. For investigations, scientific research, multimedia and different kind of simulations, it has lots of applications.

## 1.4 Contribution Summary

Previously numbers of researchers worked on 3D visualization and VR navigation separately. Their approached models are either have bulky system, complex setup and high costs [3]. These approaches limit their uses on versatile environment due to lack of portability. Besides, for 3D visualization most of the models either used sensors like Kinect sensors to achieve the depth information of the object [4]. According to the sensor based approaches the distance of the object plays a vital role. If the distance of the object is increased from the camera plane the error in finding the depth image increases. The percentage on an average of the Absolute Mean Percentage Error is 3.6% compared with the depth distance and the actual distance of the object from the camera plane [4]. In our model we have taken a more liberal approach to overcome this limitation by providing the computer generated depth image for creating a more accurate virtual stereogram.
Another thing we take into account is that for VR device navigation our proposed model uses Google’s 'Cardboard' platform to provide the structure for Head Mounted Display while the display is provided by any smartphone. Along with that, for the movement in the VR plane our model proposes a more cost effective approach which handles the movement via the accelerometer of the smartphone devices. All in all, in our proposed approach, we integrated the 3D visualization models to create a model computer generated approach for depth image and 3D visualization

technique. With addition to the 3D visualization model of stereo pairs we integrated it with the Google's Cardboard platform for VR device support and mobile's accelerometer for moving on the 3D visualized plane.

Chapter 02

# Literature Review

The advancement of technology the demand of 3D content is increasing. Therefore people are more focused on making 3D contents. The term 3D was first introduced in 1850's.In 1853 W. Rollmann 1st introduced the idea of 3D anaglyph through viewing a yellow blue drawing with red and blue glasses [5]. He also found that with a red and blue drawing the outcome was not as perfect as it is with the yellow and blue drawing because the red and blue line drawing are not visible with the yellow and blue glasses [1,6]. In 1858, Joseph D'Almeida projected the 1st 3D magic lantern where the audience wore green and red Google to view the 3D lantern [1/6]. Later on 1953, the anaglyph for 3D visualization of image spread among comics' newspapers and magazines [6].

Humans are able to have 3D effect due to their 3D vision. Human eyes perceive the same scene but from different perception because of the differences of left and right eye the scene is deviated from one to another. This visualization technique results slightly different signal to the brain [7]. Furthermore, the accuracy of the visualization improves through attributes like light, shadow etc. for better depth estimation. Mainly brain accepts horizontal disparities between two images and returns a single image with accurate depth information [7].

Many existing papers, propose techniques that provides creation of image for 3D visualization from single image camera and Kinect sensor [8]. The Kinect sensor and it's in built camera is used for capturing the color of every pixel and depth information. From this they extract the left and right image and then applied the color filtering to achieve the 3D anaglyph [9]. Moreover, there are more complex approach for estimating the depth information. There are methods were depth map is generated through 3D histogram-based segmentation to refine the depth map produced by the learning based method. Another depth extortion from single view 2D image is to measure the focus and defocus cues through higher-order statistics technique through which the foreground and background is segmented. Later on the depth map is refined from the segmented images [10]. Furthermore, high computed data mining techniques such Knn based learning are introduced for calculating the depth map [11]. The drawbacks of all these methods are they either require great computational power or not well suited for depth generation from 2D images.

Depth cues play an important role for visualization of 3D image. The human brain is a very complex system. Our eyes perceive the image from two different positions through left and right

eyes. Because of the distance between our eyes which is approximate 6.5cm this displacement of images is created which results in the stereo vision also known as stereogram [12]. Diffusion in the signals in brain gives us a 3D visualization. Recreating this visualization on computer the red-cyan techniques has been introduced for stereo images where the left eye image is filtered in red color and the right eye image is filtered in cyan and imposed on each other to recreate the 3D anaglyph. Besides, rendering one image from observer's side and applying the horizontal offset to create the stereoscopy. This offset is known as Base in stereoscopy which is assumed as 6.5cm [13]. Furthermore this method is executed when there is a pair of stereo image and depth information either collected via stereo camera and complex depth map generation techniques.
For generating stereoscopic 3D from 2D video image stitching through fundamental matrix is proposed with higher computing cost. The Homography matrix generation is considered for better result [14]. It is known that Homographies induced by the plane nTX +d= 0 under coordinates $\mathbb{\Pi}_E = (n^T,d)^T$ is [15] :

$$H_{ij} = K_i[R - tn^T/d]\, K_j^{-1} \qquad (1)$$

Again depending on the rotation of the camera the homographies may undergo different equation such as [16]

$$H_{ij}=K_iR_iR_j^TK_j^{-1} \qquad (2)$$

Furthermore, stereo images are the basis for stereoscopes. A stereoscopic device is prepared for displaying stereo pair images. Stereoscopes can either be glass oriented which was explained earlier or they can be high-end headsets. Low cost solution for this headsets are available too where they use the mobile devices as the display, with the stereoscopic image being created by two replaceable lenses. Mainly the stereo images are the basis for the headsets that provides a 3D visualization using our 3D vision.
Virtual Reality technology has been popular for a while and therefore many products have taken over the market with new techniques on VR along with other integrated features. Navigation techniques and supports have been offered in some of the products out there in market such as Oculus Rift and HTC Vive. Oculus Rift uses a pair of Oculus Sensors which tracks constellations of IR LEDs to translate movements in VR [17].
Similarly but with different technology HTC Vive offers a pair of base stations as their motion detectors. Each base station scans the room with a laser beam beams 50 times a second, alternating

between horizontal and vertical sweeps [18]. Therefore it has been a popular approach for navigation in VR to use external sensors to map movements and position of the head-mounted display. However in mobile devices the only few options of VR is provided by Samsung Gear VR, Google Cardboard, and Google Daydream which do not provide navigation movement systems with their devices but through external devices it provides the navigation [19]. Though mobile devices have sensors such as accelerometer, gyro-sensor, low range IR sensor and other sensors which can be involved in motion tracking, the services do not provide us direct access to it for movement in Virtual Space. Only the gyro-sensor is used for a head-rotating experience in VR in all the mobile. Still the lack of navigation and movement exists in mobile VR. Researches on movement in real world using mobile sensors has been done using the mentioned motion sensors [20].

Since our research is based on images in VR and through navigation we will interact with image as if it is 3D, our goals are more or less alike. Smartphone accelerometers can take readings of the device's acceleration over the 3 dimensional axis which can have many applications [21]. Accelerometer readings are not usually accurate since the gravity is always a factor when the reading is taken. Therefore in order to take linear acceleration reading the gravity must be cancelled out [22].

Therefore,

$$\text{linear acceleration} = \text{acceleration} - \text{acceleration due to gravity} \tag{3}$$

Motion processing from noisy data can be inaccurate and therefore can be solved using different techniques such as removal of noise and even integrating the noise data. But the results can drift having a large margin of error, so the results are not always promising yet it shows a possibility of making new assumptions [23].

According to the motion readings from the smartphone, the image can be processed to be given a closer view when physically moved forward and relatively further view if move backwards in a VR environment. The visual results will feel like as if it was in 3D and the objects in the 2D image will seem interactive. In the mainstream VR systems such as HTC Vive and Oculus Rift external sensors are used for navigation and virtual movements [24]. Multiple long range IR sensors, showed in Figure 1, are used track the movements of the wearer of the VR headset. Thus the navigation in virtual space can be done.

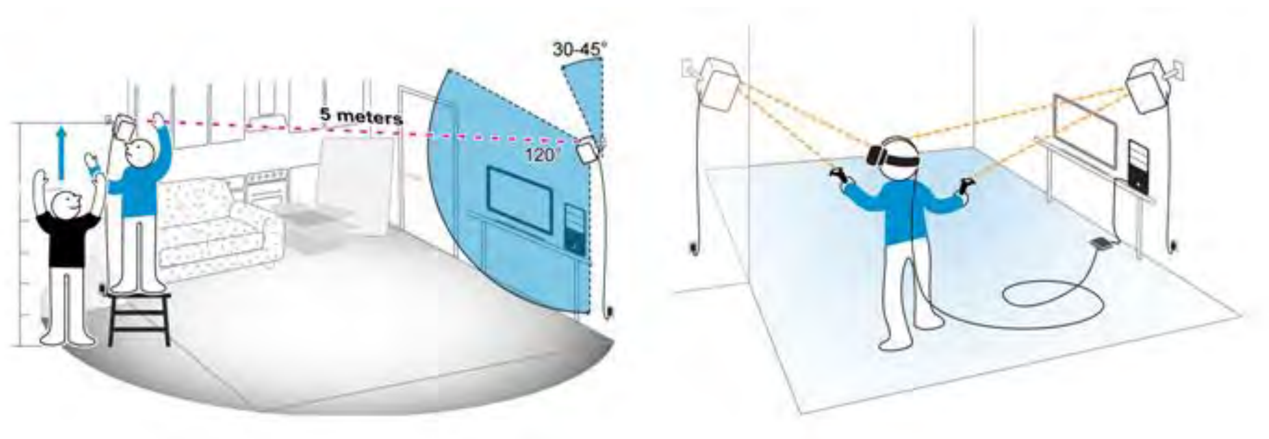


Figure 1. HTC Vive setup and navigation using base stations [25]

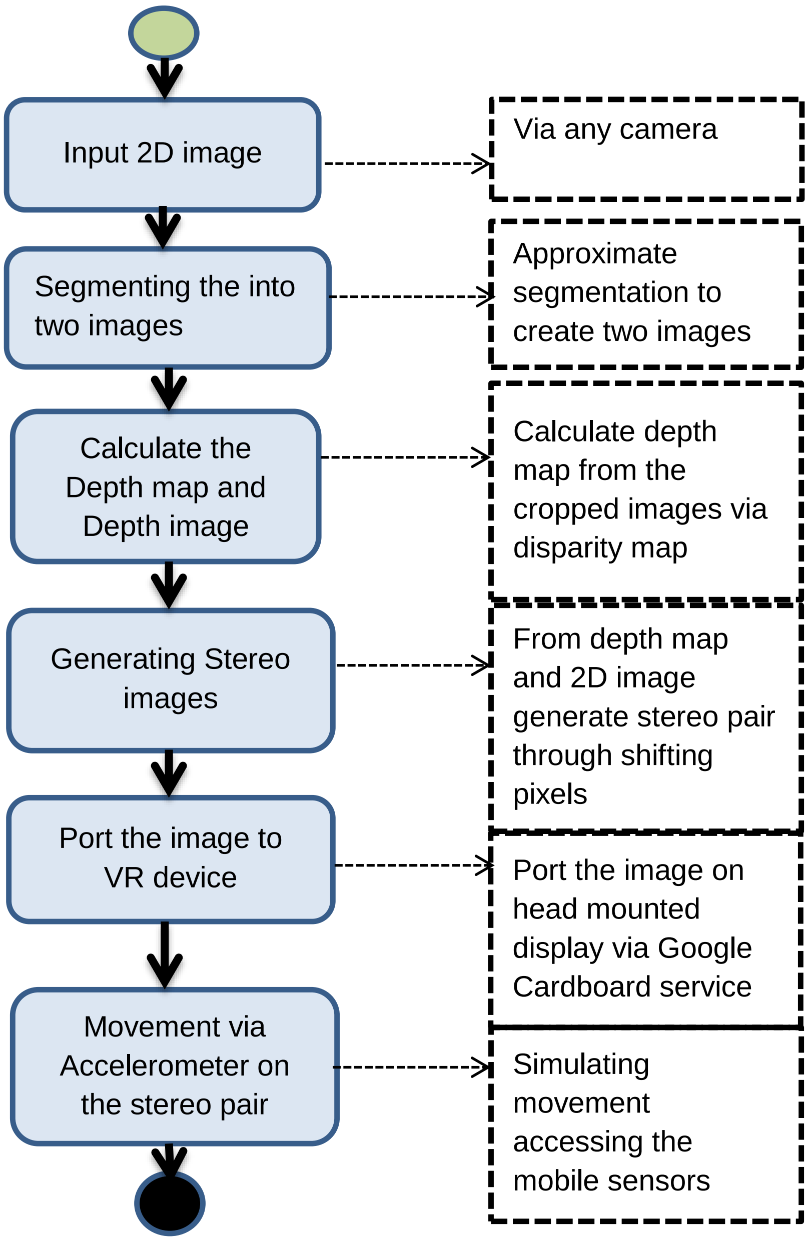
In our thesis, we are trying to create a model that gives us similar navigation results using internal sensors of smartphones rather any external sensors.

Chapter 03

# Proposed Model

## 3.1 Block Diagram of Proposed Model

Our proposed model consists for several different phases analyzing a 2D image to create a stereo pair that will be ported to the head mounted display. The basic work flow of our model is outlined on the Figure 2.



Input 2D image
Via any camera
Segmenting the into two images
Approximate segmentation to create two images
Calculate the Depth map and Depth image
Calculate depth map from the cropped images via disparity map
Generating Stereo images
From depth map and 2D image generate stereo pair through shifting pixels
Port the image to VR device
Port the image on head mounted display via Google Cardboard service
Movement via Accelerometer on the stereo pair
Simulating movement accessing the mobile sensors

Figure 2. Proposed model work flow

## 3.2 Segmentation for creating two images

Before calculating the depth map the first step that needs to be carried out is the segmentation on the image. That is the 2D image need to be divided into parts. This will be need for our further generation of the depth map. The output of the segmentation is illustrated in Figure 3

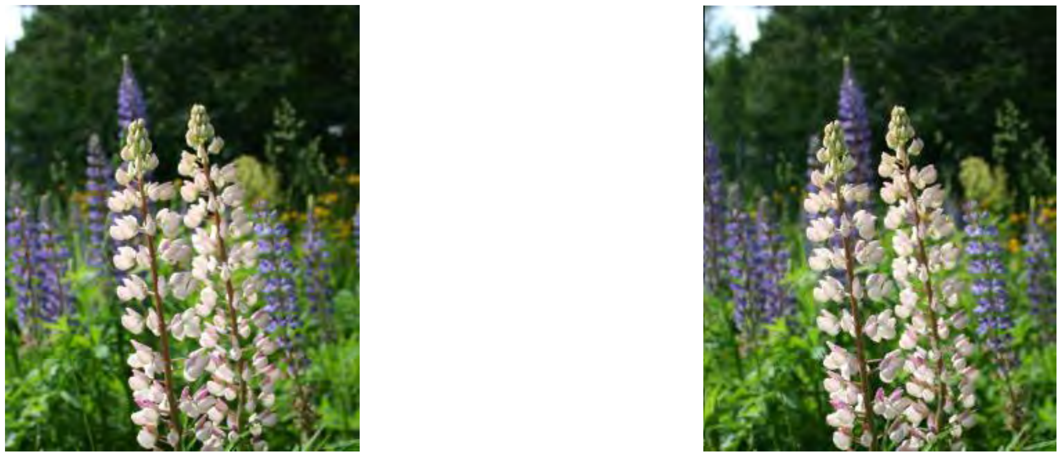


Figure 3. (a) Before segmentation. (b) After segmentation [26]

The segmentation of the image is done based on the basic human eyes anatomy. Depending on distance between the eye and the object the segmentation is done. The base has to be increased or decreased relatively to the scale of the scene to have the segmentation for two images. A typical average value for the base is 1/30 of the distance from the observer to the nearest object of a scene [29].

In the 2D image the Z axis remain constant because eyes have the same Z direction. But due to the displacement of one eye to right the horizontal (positive X axis) is displaced by the distance $T_x$, where $T_x$ is the base of the eye pair. So, the image is having (X0, Y0, Z0) coordinates values than, the segmented new pair's coordinate can be expressed as below [27].

$$N_R = (X_0 - T_x, Y_0, Z_0) \quad (4)$$

After cutting the horizontal width of the image, the image needs to be resized as the ideal image. For resizing the image, the deducted width will be converted by making each pixel of the image's x and y coordinate to color value zero. Therefore, a white space will be generated on the left corner of the image (Figure 4)

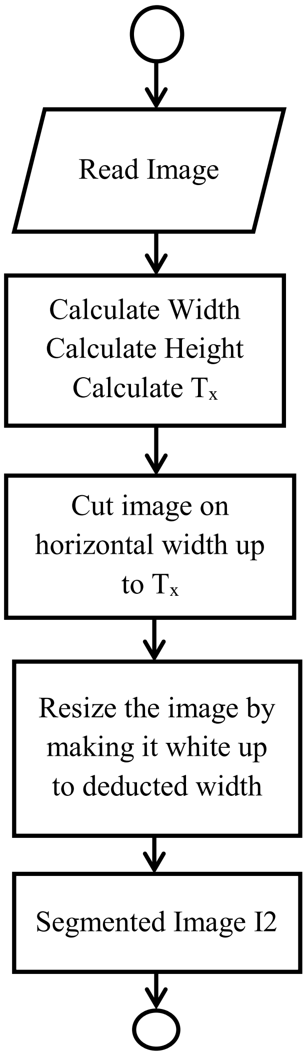

Read Image
Calculate Width
Calculate Height
Calculate $T_x$
Cut image on horizontal width up to $T_x$
Resize the image by making it white up to deducted width
Segmented Image I2

Figure 4. Flow Diagram for segmentation

## 3.3 Depth Image Generation

The next step in the proposed model, is calculating and generating the depth image. For that the new generated image and the ideal image will be taken so that the disparity can be calculated from them. Disparity happens due to the displacement of our eyes. As we are taking only a 2D image, on our previous step we created a computer generated 2 images depending on human physical environment. Therefore the difference between the horizontal x-axis will be the binocular disparity, D.

$$D = X_{idealImage} - X_{NewImage} \quad (5)$$

Moreover, our eyes are fixated on a point. Depending on it there are some conditions that was needed to be considered while calculating the disparity [27].

i. Disparity D, will hold a positive value when it is located further than the fixation point
ii. Disparity D, will hold negative value when it is located closer than the fixation point
iii. Disparity D, will be zero when the point is at the fixation point

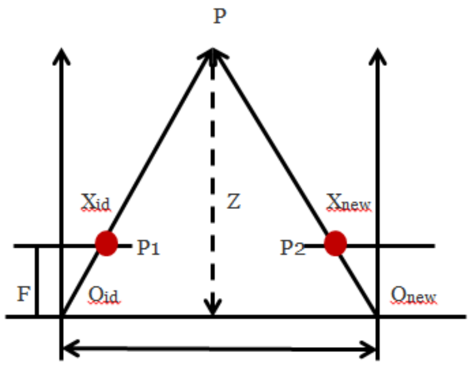


Figure 5. Disparity [27]

In the figure above, assume that P1 and P2 are the projection of the 3D point P on the ideal image and the new image. $O_{id}$ and $O_{new}$ is the coordinate of the images, f is the local length of eyes on the physical environment and Z is the depth map of the image. Based on the information above the disparity map will be calculated through Semi global block matching algorithm.
The Semi Global Block Matching algorithm which is also known as SGBM aims to minimize global energy function to obtain the disparity map. The Energy function E for disparity image, D with P2>=P1 is calculated by the following equation [28],

$$E(D) = \sum_p \left( C(\mathrm{p}, \mathrm{Dp}) + \sum_{q \in Np} P1\, I\, [|Dp - Dq| = 1] + \sum_{q \in Np} P1\, I\, [|Dp - Dq| > 1] \right) \quad (6)$$

Where

E (D) is the energy for disparity image, D

p, q represent indices for pixels in the image

Np is the neighborhood of the pixel p

$C(p, D_p)$ is the cost of pixel matching with disparity in Dp

P1 is the penalty passed by the user for a change in disparity values of 1 between neighboring pixels. P2 is the penalty passed by the user for a change in disparity values greater than 1 between neighboring pixels I is the function which returns 1 if the argument is true and 0 otherwise.

The minimized function produces a perfect disparity map with smoothing governed by parameters $P_1$ and $P_2$; however, minimizing the function for a 2D image space is an NP-complete problem. The semi-global matching function approximates the 2D minimization by performing multiple 1D, or linear, minimizations. The matching function aggregates costs on multiple paths which converge on the pixel under examination. Cost is computed for the disparity range specified by the minimum disparity and number of disparities parameters. By default, the matching algorithm aggregates costs for 5 directions. By setting the parameter, it can be forced to aggregate costs for 8 directions.

Let, $S(p, d)$ be the aggregate cost for pixel $p$ and disparity $d$. Then [28]

$$S(p, d) = \sum_r Lr(p, d) \tag{7}$$

Where,

r is a direction used for converging to the pixel p

$L_r(p, d)$ is the minimum cost of the path taken in direction r from pixel p for disparity d

The cost $L_r(p, d)$ is given in the following equation:

$$L_r(p,d)=C(p, d)+\min(L_r(p-r,d),L_r(p-r,d-1)+P_1 . L_r(p-r, d+1) +P_1 . \min_i L_r(p-r ,k)+P_2)-\min_k L_r(pr,k) \tag{8}$$

The equation uses the following costs to find the disparity by adding current cost, C (p, d, to previous pixel in direction r [28]:

• The minimum of the cost at previous pixel with disparity d

• The cost at previous pixel with disparity d - 1 and d + 1 with added penalty P1

• The cost at previous pixel with disparities less than d - 1 and greater than d + 1 with added penalty P2

In order to limit the ever increasing value of $L_r(p, d)$ on the path, minimum value of the previous pixel is subtracted. The upper value of $L_r(p, d)$ is bounded by $C_{max}$ + P2, where $C_{max}$ is the maximum value of cost C. The cost function C (p, d) is computed in the following manner [28]

$$C(p,d)= \min(d(p,p-d,I_L,I_R), d(p-d, p, I_R,I_L)) \tag{9}$$

Where,

IL and IR are left and right rectified images, respectively [28]

$$d\ (p,p\text{-}d,\ I_L\ ,\ I_R) = \min_{p-d-0.5 \leq p-d+0.5}\ |\ I_L\,(p) - I_R\,(q)\ | \qquad (10)$$

The value of C is aggregated over a window of a user-defined size. After computing S (p, d) for each pixel p for each disparity d, the algorithm chooses the disparity which provides the minimum cost for that pixel.

Furthermore, the SGBM algorithm generates the disparity. From the disparity the depth image is generated by given formula [27]

$$Z = f * \left(\frac{T}{d}\right) \qquad (11)$$

Where,

Z is the depth to be calculated, f is the focal length, T is the baseline & d is the disparity.

The overall work flow for generating the depth map is summarized in the Figure 6.

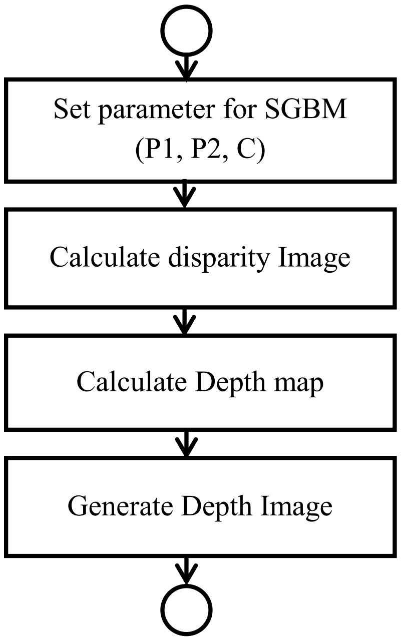


Figure 6. Generating Depth Image Work Flow

## 3.4 Stereo-pair Generation

The viewer distance from the screen plays vital role. The human vision aperture is 46 degrees in diagonal therefore the comfortable vision for the screen is evaluated. If the distance D (Figure 7), between the screen and the viewer is equal to the max depth effect into the Screen P, the achieved parallax is equal to B/2 thus more comfortable than B [29].

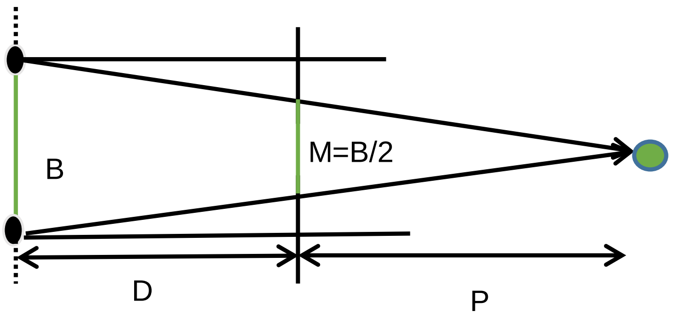


Figure 7. Parallax relation with respect to screen

For synthesizing the left and right images, the model proposes computing each of the pixel's parallax value from the estimated depth map. After that shift each pixel by the corresponding parallax values in an input image. The parallax value at (x, y), Parallax (x, y) is computed from the depth map as follows [29, 10].

$$\text{Parallax}(x, y) = M * (1 - \frac{depth(x,y)}{255}) \qquad (12)$$

Where M denotes the maximum parallax value and depth (x, y) is the estimated depth value at (x, y).

For every pixel of the input image, the parallax is calculated from the estimated depth map. Considering the input image as the virtual central view from which the left and right image are obtained (Figure 8) by shifting the input pixels by a value equal to the *parallax/2* for each view [29, 10].

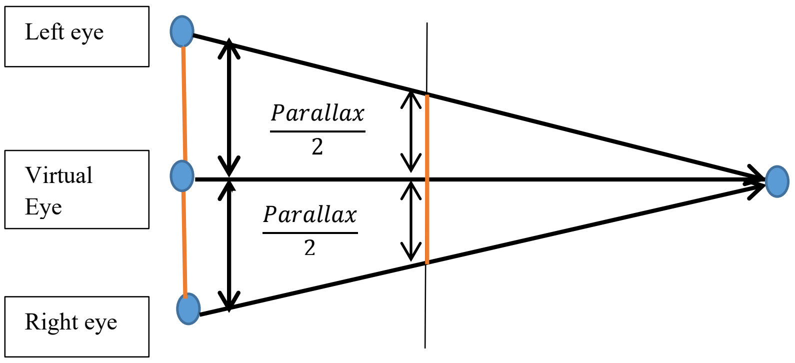
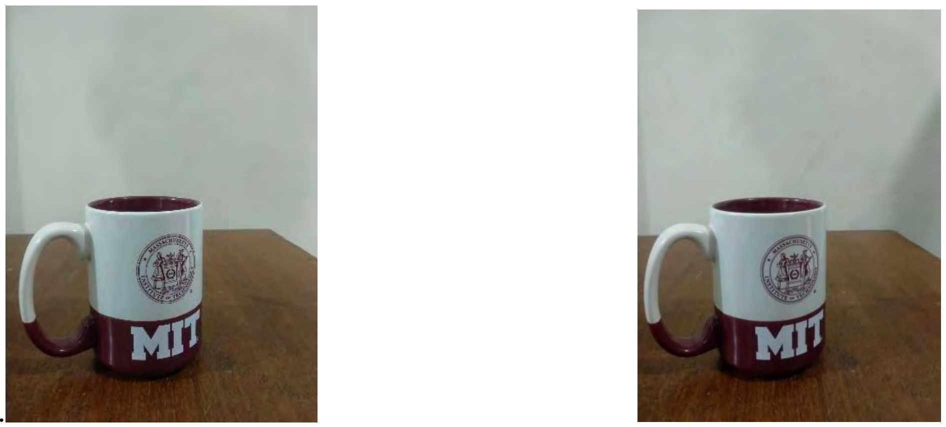


Figure 8. Stereo-pair generation

As shown on the figure no. 3.7 the image is considered as the center of the stereoscopic image pair. Here shifting of each pixel of the input image by the amounts of parallax (x, y)/2 to the left direction to generate the left view image (Figure 9) [29]. Similarly, right viewed image can be achieved through the process (Figure 9).



(a) (b)

Figure 9. (a) Left eye image (b) right eye image

## 3.5 Implementation using Google Cardboard on Android Studio

The computer generated stereo pair then will be implemented in the mobile device using Google Cardboard SDK and the IDE Android Studio or Unity3D. Both the IDE has different methods to import stereo image pair. The Android Studio usually uses Cardboard Camera image or any 360 image (Figure 10) converted into Cardboard Camera image to use in panorama view (image viewer for panorama) (Figure 11).

Figure 10. 360 image

The Cardboard Camera image is a specific image pattern used to port a panorama image to panorama view using Android Studio. The left image and right image is placed up and down within the same distance of the original image's height. This procedure step is specifically done to port images to Android Studio. However, images captured Cardboard Camera can be directly ported since it already has the Cardboard support and can format while porting automatically.

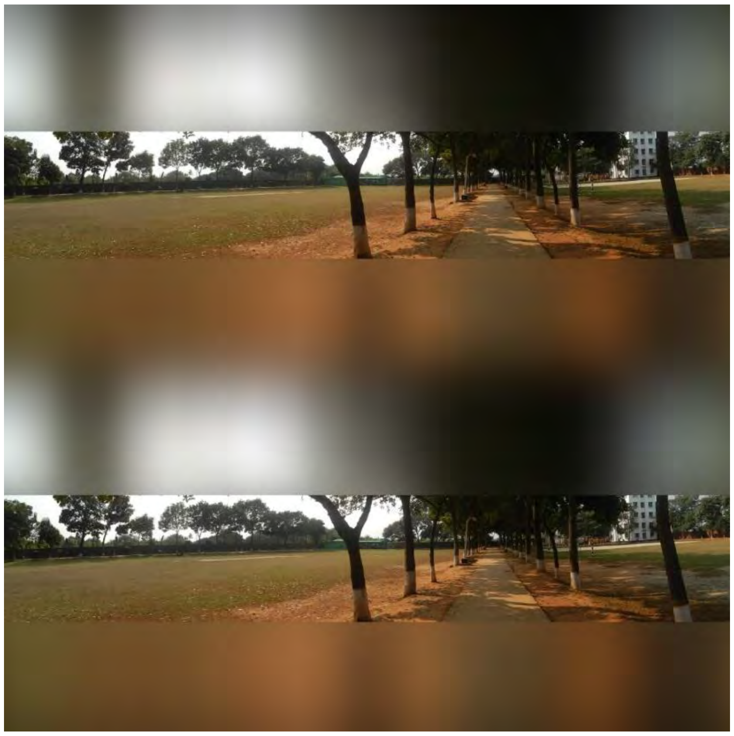


Figure 11. Cardboard image using Cardboard camera converter tool

Cardboard camera image can also be generated using the Cardboard camera tool provided by Google [30]. Then the image can be directly imported in the android application through VR Panorama View in Figure 3.12 [31].

The Panorama View has the option to view the imported image in VR view so that it can be directly used using the physical Google Cardboard headset [32].

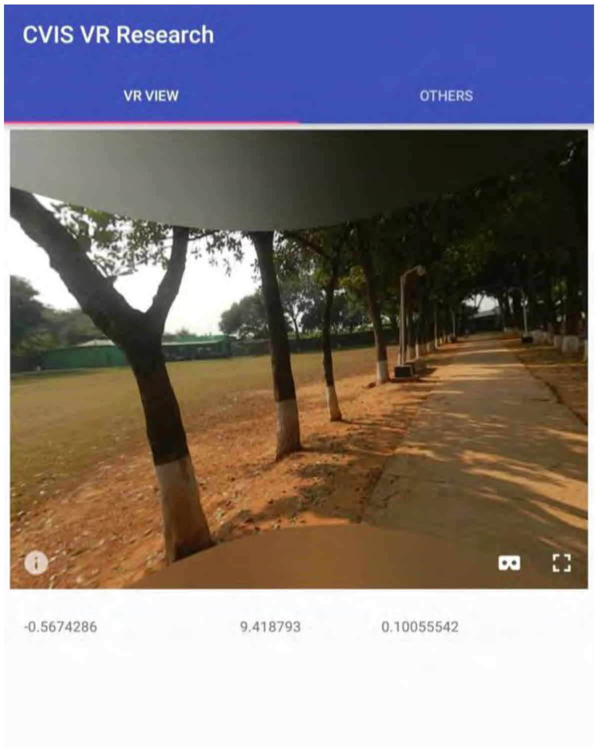


Figure 12. VR Panorama View interface

Similarly for IOS phones Panorama View can be implemented too [33]. After that the VR view can be executed.

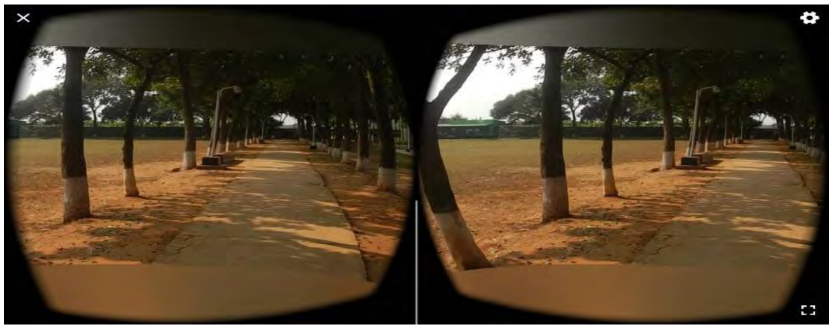


Figure 13. VR View using Android Studio

Additionally, the Google Cardboard SDK for Android Studio (Figure 13) allows the viewer to access the gyro sensor to move the head and to experience surround effect.

## 3.6 Implementation using Google Cardboard on Unity3D

The same results can be get using Google Cardboard SDK for Unity3D. The process differs since the SDK is created for each IDE's different mechanisms. Unity3D deals with three-dimensional objects rather than images. Therefore the VR view made in Unity3D is different but resulting mostly similar.

Firstly we took a Sphere 3D object and characterized the 'Shader' in 'Flipping Normals' as shown in Figure 14. This will make any image on the sphere inside out, displaying the imported texture on the sphere from inside [34].

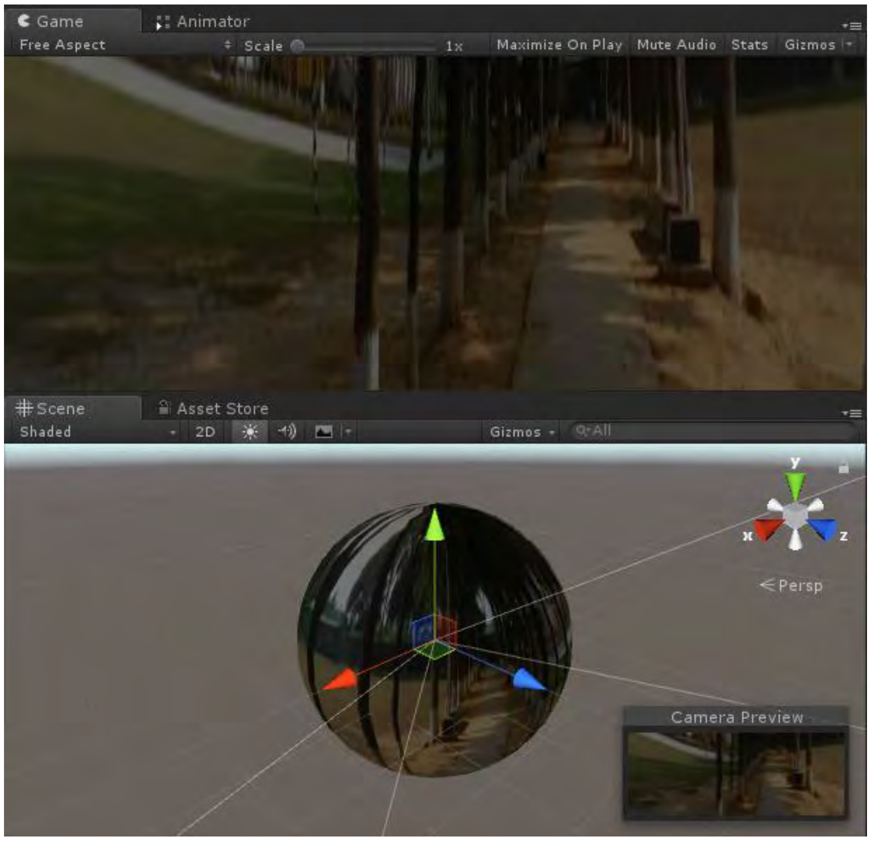


Figure 14. Developing the application for VR view in Unity3D

The camera position remains inside the sphere giving the view of the image which is imported as the spheres texture from the inside (Figure 14). Now, the SDK allows the camera to view the image in 360. Even without a 360 photo this effect can be achieved by scaling down the image as the texture for the sphere and doing the rest of the previously mentioned procedure (Figure 23).

Additionally, the Google Cardboard SDK for Unity3D allows the viewer to have 360 surround effect by rotating their head. Thus the sphere 3D object is used so that the viewer is surrounded by the image in Figure 15.

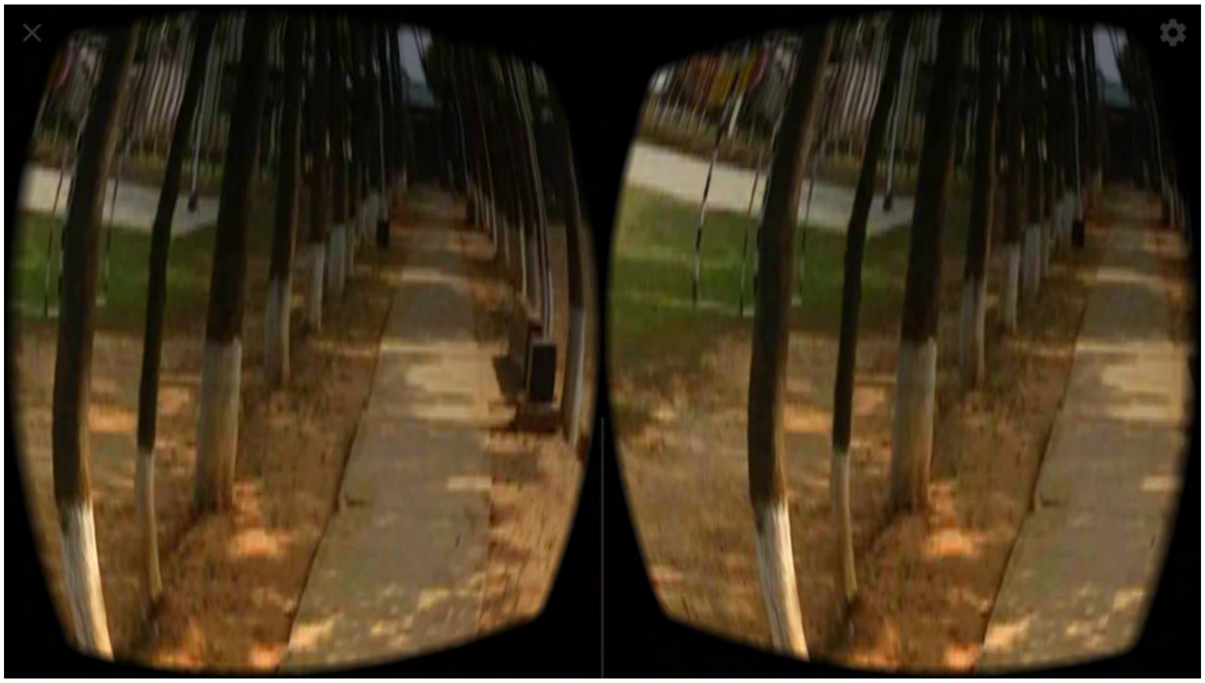


Figure 15. VR view using Unity3D

After running it on VR view using Unity3D, the visual will look like Figure 3.14 giving the surrounding effect to the viewer.

## 3.7 Motion reading and processing

The motion detection using accelerometer is similar in both Android Studio and Unity3D since both the IDE fetches the sensor reading from the hardware.

In Android Studio the sensor reading are managed through Sensor Manager. Sensor Manager fetches the information of the sensor which is used to take readings from all the available sensors of the android device [35]. Accelerometer reading for all the three axis x, y and z can be fetched and used.

The three elements of the rotation vector are expressed in Figure 16

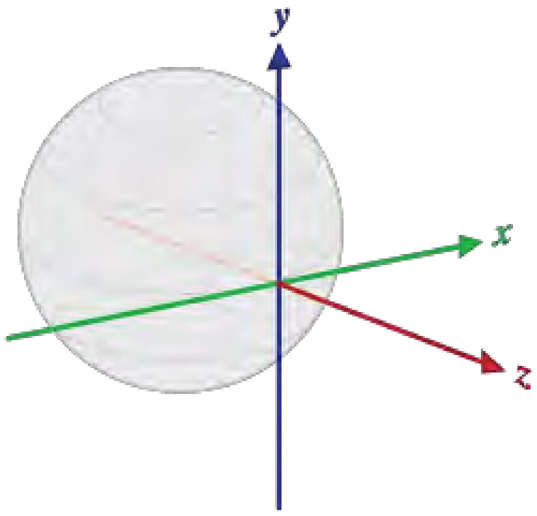


Figure 16. Coordinate system used by the rotation vector sensor

The equation the follows these rotation vectors are

$$x.\sin(\frac{\theta}{2}) \tag{13}$$

$$y.\sin(\frac{\theta}{2}) \tag{14}$$

$$z.\sin(\frac{\theta}{2}) \tag{15}$$

Where the magnitude of the rotation vector is equal to (13), and the direction of the rotation vector is equal to the direction of the axis of rotation.

The three elements of the rotation vector are equal to the last three components of a unit quaternion

$$(cos\left(\frac{\theta}{2}\right), x*\sin(\frac{\theta}{2}), y*\sin(\frac{\theta}{2}), z*\sin\frac{\theta}{2}) \tag{16}$$

Elements of the rotation vector are unitless. The x, y, and z axes are defined in the same way as the acceleration sensor [22].

In Unity3D accelerometer reading is taken directly rather than calling other context to fetch sensor services. In our experiment we used the z-axis value to determine the forward and backward movement of the device [36].

The taken accelerometer readings do not give linear acceleration data since the reading involves gravity, tilt noises etc. So in order to get perfect linear acceleration we have to get rid of the gravity (3).

Linear acceleration = acceleration - acceleration due to gravity (3)

Since our target is to get linear movement in forward and backwards, we will be taking acceleration over z-axis.

Therefore,

$$a_{linearZ} = A - g - Noise \quad (17)$$

$$Noise = A - g \text{ (at rest)} \quad (18)$$

Where, $$A = a_x + a_y + a_z \quad (19)$$

$$g = 9.8 \text{ ms}^{-2}, \quad Noise \sim 2 \text{ ms}^{-2} \text{(experimental)}$$

Using the equation we can estimate the linear acceleration over z-axis to determine the amount of movement on forward and backward.

The process of showing the shrinking effect in Android Studio is by using Java OpenGL where the image can shrink each time the forward acceleration occurs.

In Unity3D it is simpler as the acceleration data can be directly used to translate the 3D sphere closer to the virtual camera object.

Chapter 04

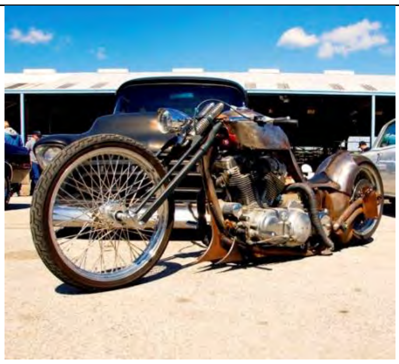
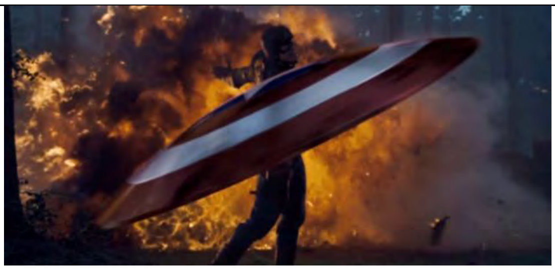
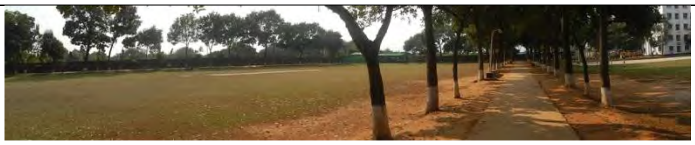
# Experimental Result and Discussion

## 4.1 Experimental Result

The proposed model is simulated has been simulated for extracting the results. Table 4.1 shows the input 2D images.

Different sets of 2D input Images collected via camera (Figure 4.18), 2D movie (Figure 4.17) and a random image from the internet (Figure 4.16).

Table1. Different sample images for experimentation

| Input 2D Image type | Images |
|---|---|
| 2D Image [37] |  Figure 17. Input 2D image |
| 2D Movie Scene [38] |  Figure 18. Input 2D image from movie scene |
| 360° image |  Figure 19. Input 360 panorama image |

The images shown in Table1 will be segmented to create two images taking the input image as ideal and creating the segmented one using our technique.

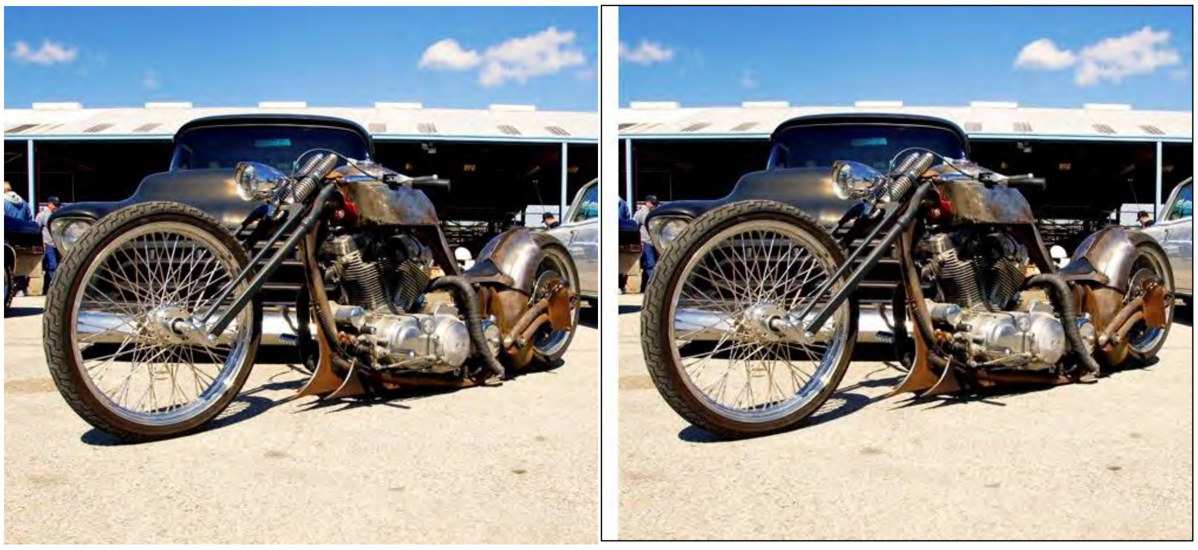

(a) (b)

Figure 20. (a) 2D ideal image (b) segmented images of 2D image

In Figure 20, segmented image is created. The ideal (Figure 20 (a)) is segmented accordingly and the result is the segmented 2D image (Figure 20 (b)). According to our proposed plan the image is then resize through creating the white space up the segmented part.

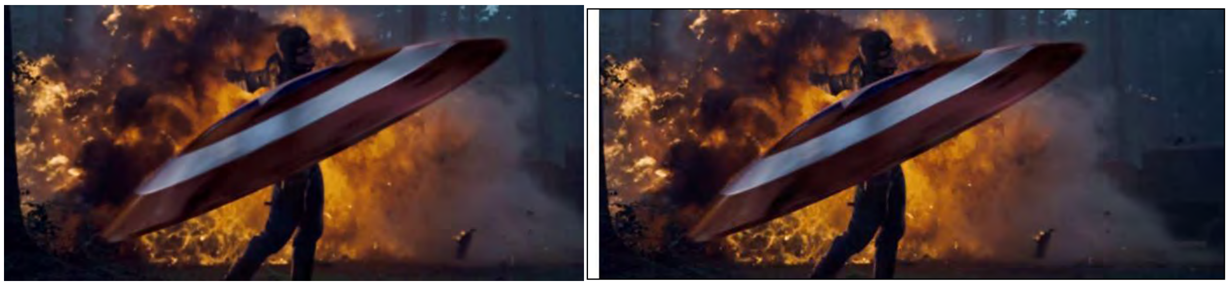
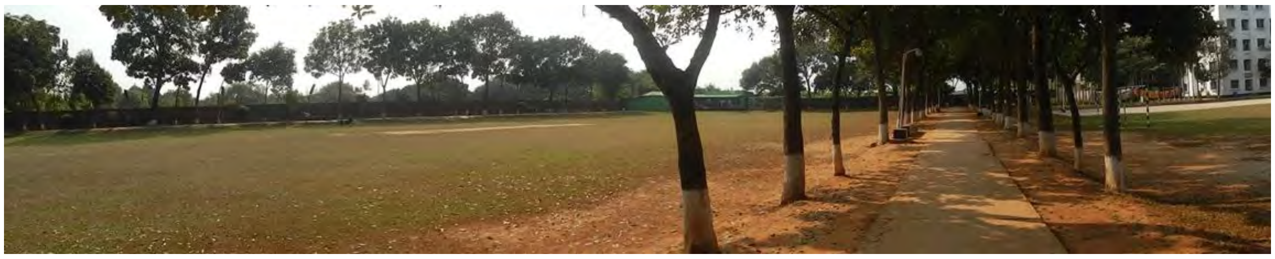
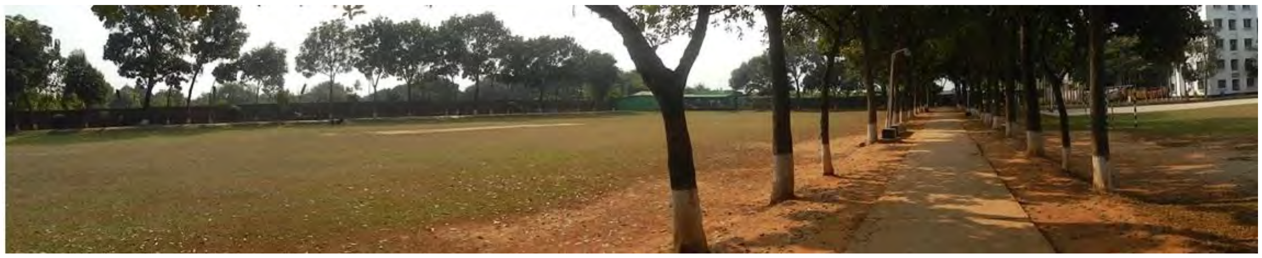

(a) (b)

Figure 21. (a) 2D Movie ideal image (b) segmented image

In Figure 21, a 2D movie image (Figure 18) is segmented. To make the segmented image as the ideal image it has been resized accordingly to the proposed methodology (Figure 21 (b)).



**(a)**

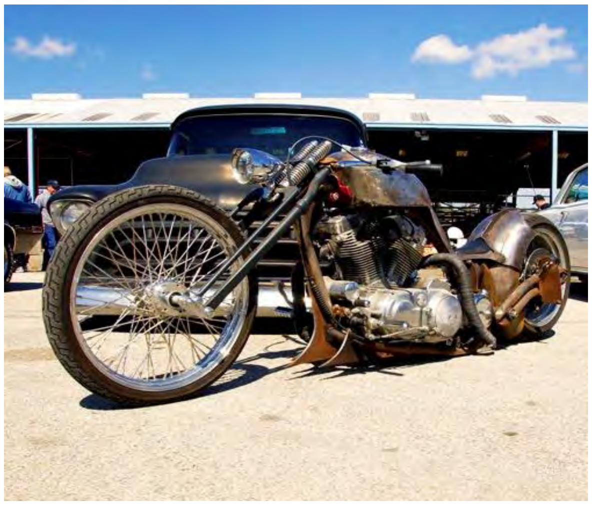


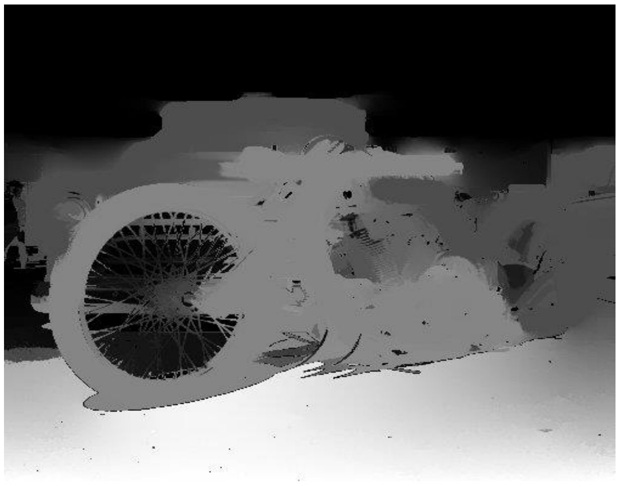
**(b)**

Figure 22. (a) $360^0$ ideal image (b) segmented image

Because of the VR navigation, the $360^0$ image was captured via a camera (Figure 19) and then it is segmented with the white space in it (Figure 22 (b)).

In Figure 23, the depth of the image is created. Firstly the disparity map is done through the semi global block matching algorithm and then the depth map is calculated (11). The Figure 4.23 (b) is the result of the depth map that has been created by the ideal and segmented images (Figure 4.20).





**(a)** **(b)**

Figure 23. (a) 2D image (b) Depth Image

From the ideal and segmented images of Figure 21, the depth image of Figure 24 (b) is generated by the method.

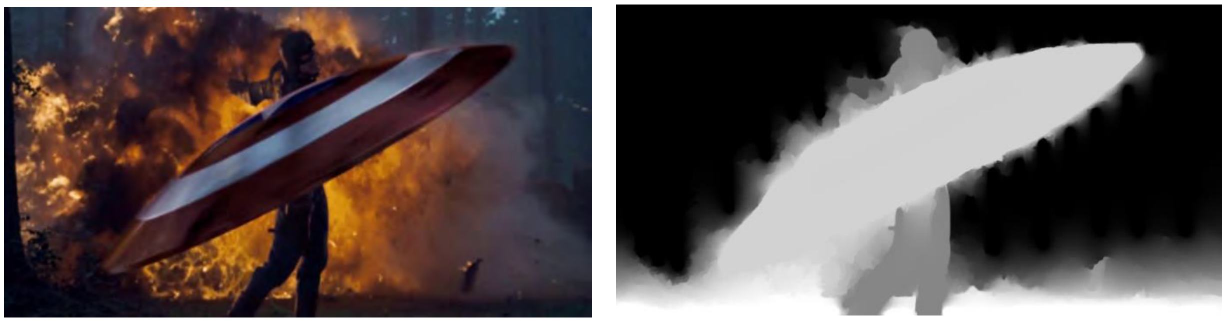
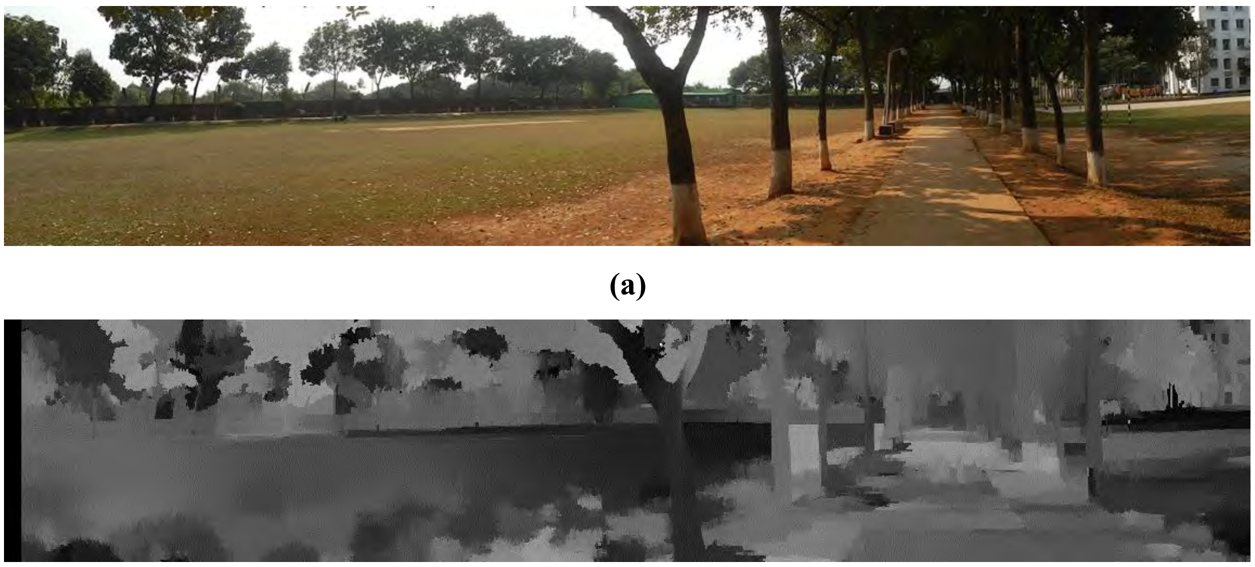

**(a)** **(b)**

Figure 24. (a) 2D movie image (b) Depth Image

The $360^0$ segmented ideal image goes through the SGBM algorithm and the depth map equation (12) to produce the depth image (Figure 25).

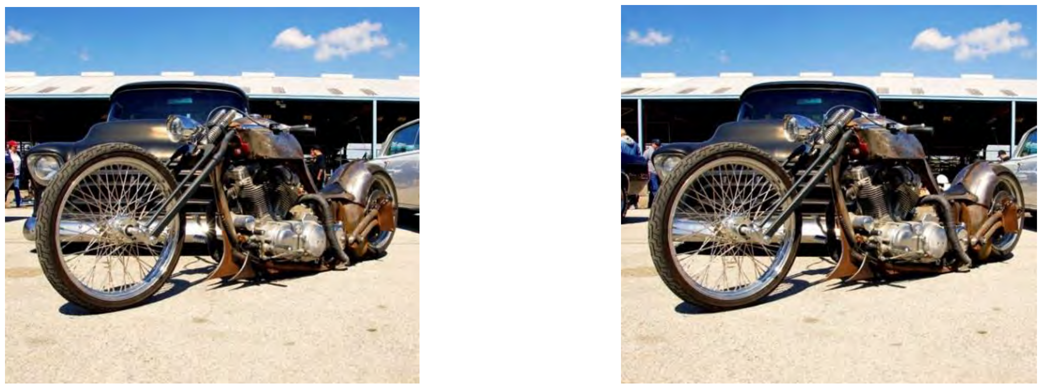

**(a)**

**(b)**

Figure 25. (a) $360^0$ image (b) Depth Image

After the generation of computer generated depth image, based on the ideal image and the depth image, the stereo pairs, that is the left eye image and right eye image is generated by the equation (12).

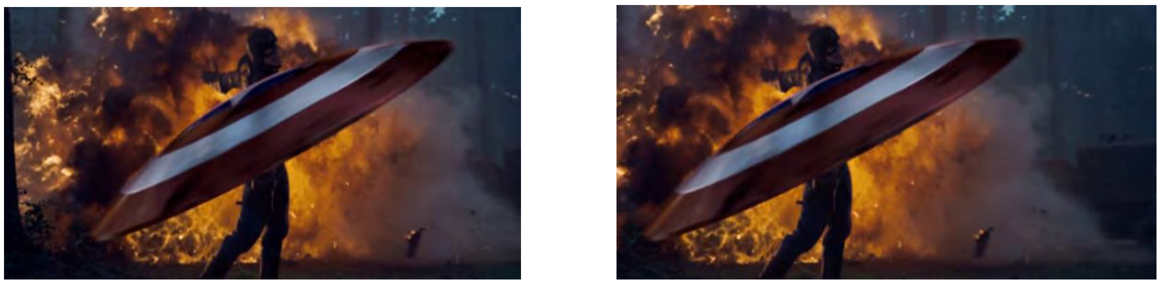
In Figure 26, the left eye and the right eye image is created from the given image (Figure 17). This stereo pairs are then ported for VR navigation.



**(a)** **(b)**

Figure 26. (a) Left eye image (b) Right eye Image

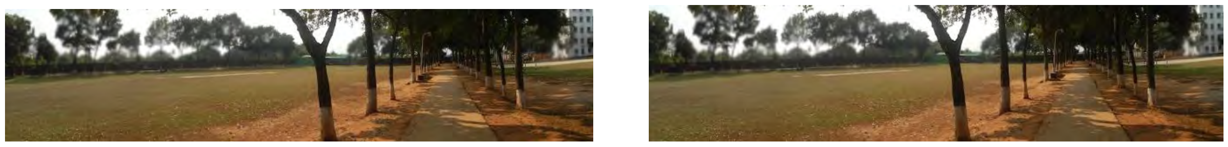
From 2D image of the movie scene, the stereo pair image (Figure 27) is generated with the help of the depth map and the ideal image, according to our model.



**(a)** **(b)**

Figure 27. (a) Left eye image (b) Right eye Image

Figure 28, represents the stereo image of the $360^0$ image (Figure 17) that has been created from the proposed methods.



**(a)** **(b)**

Figure 28. Stereo Pair (a) Left eye image (b) Right eye Image

The acceleration reading using android mobile device was taken to plot a graph (Figure 29) to show the behavior of the acceleration on the three axis. The accelerometer reading may differ from device to device. The smart phone device was in landscape mode while determining the acceleration movement.

The following data (Table 2) was taken from the smart phone device while moving forward. The data is collected through the sensor manager of smartphone devices.

Table 2. Reading of Accelerometer for forward movement

| Time | dataX | dataY | dataZ | Time | dataX | dataY | dataZ |
|---|---|---|---|---|---|---|---|
| 0.1 | 0 | 0 | 0 | 3.4 | 8.785 | -1.035 | 4.811 |
| 0.2 | 9.289 | -0.778 | -0.868 | 3.5 | 8.804 | -1.074 | 4.09 |
| 0.3 | 9.339 | -0.747 | -0.808 | 3.6 | 8.993 | -1.012 | 3.849 |
| 0.4 | 9.256 | -0.742 | -0.904 | 3.7 | 9.114 | -0.908 | 2.859 |
| 0.5 | 9.306 | -0.719 | -0.804 | 3.8 | 9.165 | -1.148 | 2.415 |
| 0.6 | 9.309 | -0.723 | -0.715 | 3.9 | 9.107 | -0.361 | 2.332 |
| 0.7 | 9.325 | -0.753 | -0.892 | 4 | 8.903 | -0.564 | 2.619 |
| 0.8 | 9.282 | -0.74 | -0.956 | 4.1 | 8.747 | -0.129 | 1.883 |
| 0.9 | 9.261 | -0.722 | -1.048 | 4.2 | 8.88 | -0.137 | 1.11 |
| 1 | 9.203 | -0.723 | -1.059 | 4.3 | 8.97 | -0.609 | 1.422 |
| 1.1 | 9.321 | -0.681 | -0.949 | 4.4 | 9.122 | -0.301 | 1.135 |
| 1.2 | 9.33 | -0.728 | -0.944 | 4.5 | 9.238 | -0.294 | 1.25 |
| 1.3 | 9.312 | -0.632 | -0.859 | 4.6 | 9.145 | -0.53 | 0.867 |
| 1.4 | 9.261 | -0.63 | -0.93 | 4.7 | 9.343 | -0.7 | 0.825 |
| 1.5 | 9.261 | -0.687 | -1.077 | 4.8 | 9.385 | -0.601 | 1.161 |
| 1.6 | 9.225 | -0.629 | -1.04 | 4.9 | 9.449 | -0.46 | 1.116 |
| 1.7 | 9.233 | -0.692 | -0.876 | 5 | 9.133 | -0.75 | 1.359 |
| 1.8 | 9.227 | -0.746 | -0.986 | 5.1 | 9.157 | -0.943 | 1.562 |
| 1.9 | 9.338 | -0.881 | -0.759 | 5.2 | 9.154 | -0.908 | 1.163 |
| 2 | 9.343 | -0.859 | -0.523 | 5.3 | 9.267 | -0.864 | 1.429 |
| 2.1 | 9.443 | -0.729 | -0.314 | 5.4 | 9.222 | -0.902 | 1.482 |
| 2.2 | 9.469 | -0.622 | -0.551 | 5.5 | 9.241 | -0.886 | 1.356 |
| 2.3 | 9.354 | -0.596 | -0.345 | 5.6 | 9.201 | -0.845 | 1.184 |
| 2.4 | 9.1 | -0.53 | -1.245 | 5.7 | 9.172 | -0.767 | 1.127 |
| 2.5 | 9.211 | -0.372 | -1.298 | 5.8 | 9.209 | -0.749 | 1.267 |
| 2.6 | 8.918 | -0.349 | -1.109 | 5.9 | 9.284 | -0.843 | 1.218 |
| 2.7 | 8.939 | -0.325 | -0.719 | 6 | 9.234 | -0.788 | 1.247 |
| 2.8 | 9.357 | -0.26 | -0.188 | 6.1 | 9.279 | -0.822 | 1.112 |
| 2.9 | 9.171 | -0.34 | 0.044 | 6.2 | 9.281 | -0.935 | 1.149 |
| 3 | 10.023 | -0.923 | 0.296 | 6.3 | 9.258 | -0.992 | 1.12 |
| 3.1 | 8.297 | -1.016 | 4.109 | 6.4 | 9.241 | -0.86 | 1.186 |
| 3.2 | 7.835 | -0.806 | 5.954 | 6.5 | 9.279 | -0.919 | 1.189 |
| 3.3 | 8.8 | -0.981 | 6.136 | 6.6 | 9.283 | -0.911 | 1.1 |

After plotting the values of the accelerometer sensor reading in respect of time, it shows a graph of forward movement acceleration (Figure 29).

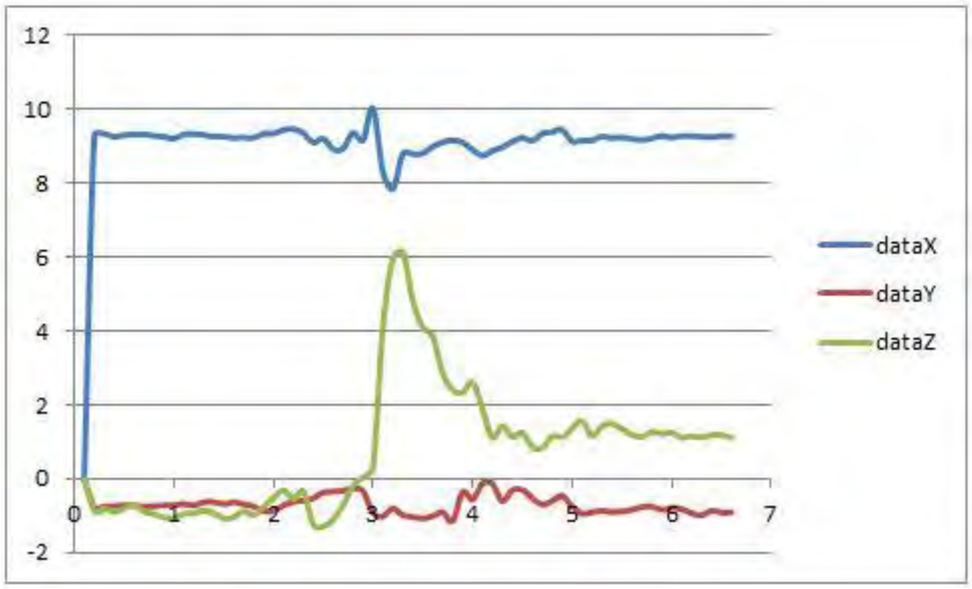


Figure 29. Forward movement acceleration on graph

The spike on the Z-axis on Figure 29 data indicates the physical forward movement of the headset. Therefore for each forward movement there will be a positive spike on the Z-axis.

In the Figure 30, according to the movement reading from the table the image will be shrunk with transitions for better experience.

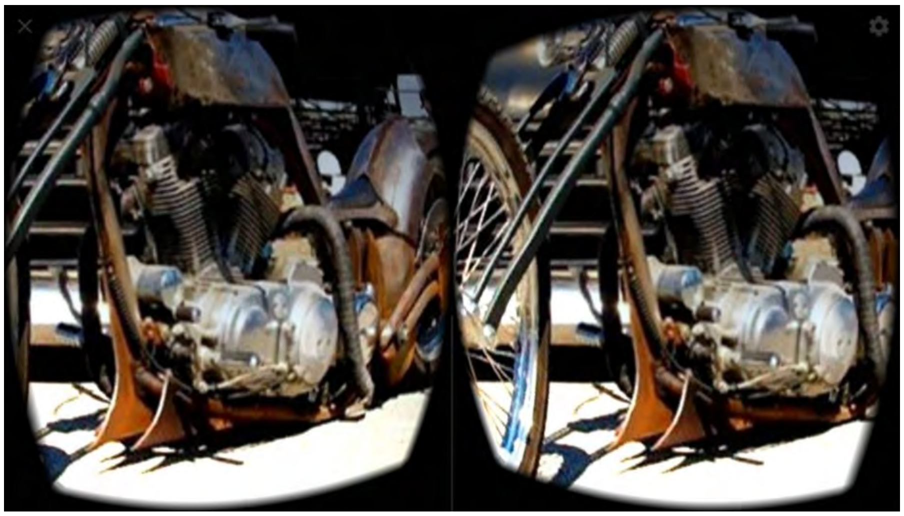


Figure 30. 2D (I) Sample Image in VR view navigating forward

Here similarly, a frame from the Figure 31 shows while moving forward the objects seem closer to the viewer.

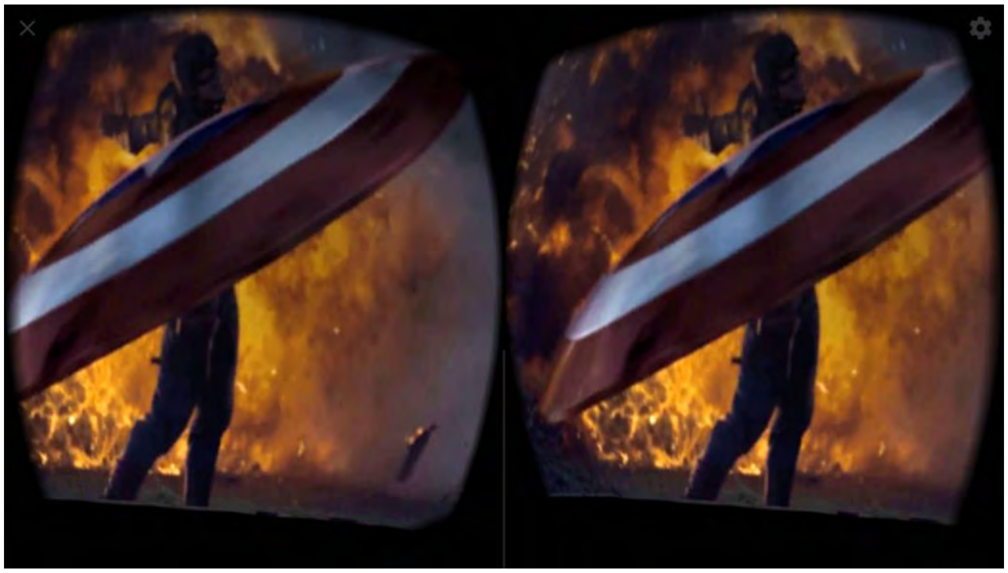


Figure 31. 2D Sample Image (II) in VR view navigating forward

Also, with a 360 image of Figure 19, the navigation will feel more natural and moving forward physically will give a sense of moving forward in the virtual space as well showed in Figure 32.

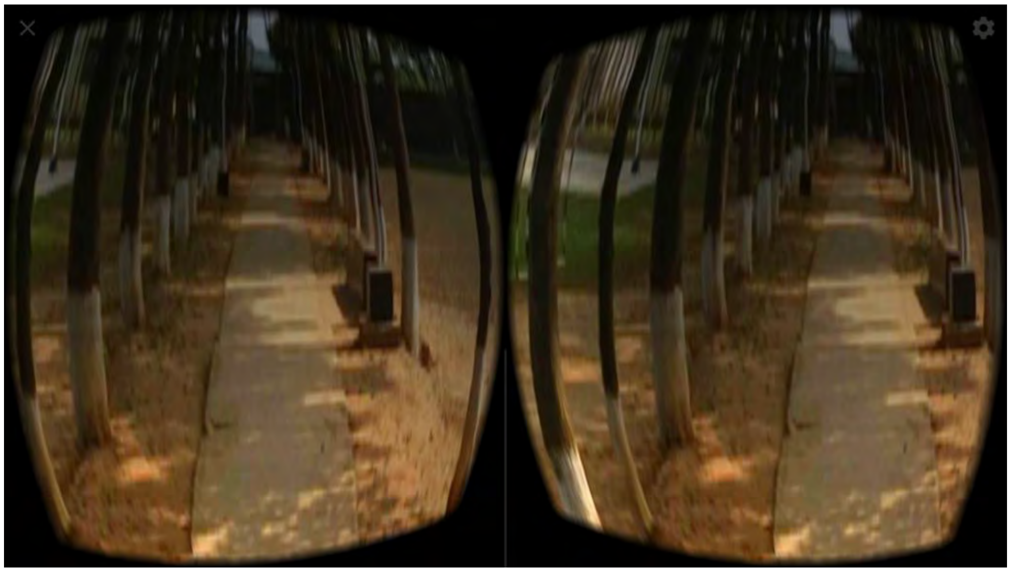


Figure 32. 360 Sample Image in VR view navigating forward

For backward movement we took similar data via sensor manager on every axis in Table 4.3 and plotted graph in Figure 33 to analyze the behavior change.

Table 3. Reading of Accelerometer for backward movement

| Time | dataX | dataY | dataZ | Time | dataX | dataY | DataZ |
|---|---|---|---|---|---|---|---|
| 0.1 | 0 | 0 | 0 | 3.7 | 8.918 | -1.035 | -1.634 |
| 0.2 | 9.316 | -0.693 | 0.753 | 3.8 | 8.968 | -0.756 | -2.508 |
| 0.3 | 9.31 | -0.667 | 0.947 | 3.9 | 9.022 | -0.696 | -2.339 |
| 0.4 | 9.261 | -0.655 | 0.643 | 4 | 9.055 | -0.697 | -2.081 |
| 0.5 | 9.269 | -0.628 | 0.902 | 4.1 | 9.148 | -0.977 | -1.454 |
| 0.6 | 9.22 | -0.553 | 0.882 | 4.2 | 9.398 | -0.921 | -1.243 |
| 0.7 | 9.273 | -0.487 | 0.731 | 4.3 | 9.209 | -0.666 | -0.988 |
| 0.8 | 9.256 | -0.499 | 0.861 | 4.4 | 9.192 | -0.672 | -0.752 |
| 0.9 | 9.249 | -0.463 | 0.885 | 4.5 | 9.204 | -0.613 | -0.595 |
| 1 | 9.293 | -0.517 | 0.847 | 4.6 | 9.017 | -0.464 | -1.107 |
| 1.1 | 9.288 | -0.526 | 0.961 | 4.7 | 9.099 | -0.209 | -1.065 |
| 1.2 | 9.307 | -0.552 | 0.929 | 4.8 | 9.255 | -0.301 | -0.839 |
| 1.3 | 9.264 | -0.57 | 1.004 | 4.9 | 9.328 | -0.283 | -0.436 |
| 1.4 | 9.246 | -0.617 | 0.89 | 5 | 9.436 | -0.562 | -0.434 |
| 1.5 | 9.266 | -0.518 | 1.074 | 5.1 | 9.266 | -0.625 | -0.28 |
| 1.6 | 9.229 | -0.514 | 1.152 | 5.2 | 9.34 | -0.842 | -0.191 |
| 1.7 | 9.246 | -0.603 | 1.11 | 5.3 | 9.448 | -1.007 | -0.453 |
| 1.8 | 9.295 | -0.659 | 1.3 | 5.4 | 9.203 | -0.958 | -0.49 |
| 1.9 | 9.141 | -0.543 | 1.218 | 5.5 | 9.23 | -0.999 | -0.378 |
| 2 | 9.204 | -0.601 | 1.017 | 5.6 | 9.334 | -1.026 | -0.359 |
| 2.1 | 9.096 | -0.429 | 1.144 | 5.7 | 9.254 | -1.075 | -0.219 |
| 2.2 | 9.092 | -0.346 | 1.039 | 5.8 | 9.212 | -0.958 | -0.041 |
| 2.3 | 9.13 | -0.186 | 1.211 | 5.9 | 9.231 | -0.948 | 0.059 |
| 2.4 | 9.406 | -0.279 | 1.443 | 6 | 9.271 | -0.924 | 0.229 |
| 2.5 | 9.416 | -0.389 | 1.652 | 6.1 | 9.358 | -0.721 | 0.509 |
| 2.6 | 9.272 | -0.401 | 1.712 | 6.2 | 9.292 | -0.648 | 0.768 |
| 2.7 | 9.187 | -0.271 | 1.393 | 6.3 | 9.377 | -0.83 | 0.753 |
| 2.8 | 9.091 | -0.303 | 1.526 | 6.4 | 9.266 | -0.745 | 0.594 |
| 2.9 | 9.224 | -0.152 | 2.211 | 6.5 | 9.21 | -0.568 | 0.679 |
| 3 | 9.306 | -0.277 | 2.223 | 6.6 | 9.333 | -0.477 | 0.843 |
| 3.1 | 9.26 | -0.246 | 1.569 | 6.7 | 9.26 | -0.651 | 0.825 |
| 3.2 | 9.166 | -0.622 | 0.598 | 6.8 | 9.32 | -0.731 | 0.849 |
| 3.3 | 9.284 | -1.023 | -0.046 | 6.9 | 9.24 | -0.763 | 0.927 |
| 3.4 | 9.451 | -1.042 | -1.238 | 7 | 9.252 | -0.737 | 0.888 |
| 3.5 | 9.695 | -1.212 | -1.363 | 7.1 | 9.234 | -0.753 | 0.827 |
| 3.6 | 9.315 | -0.943 | -2.023 | 7.2 | 9.233 | -0.631 | 0.884 |

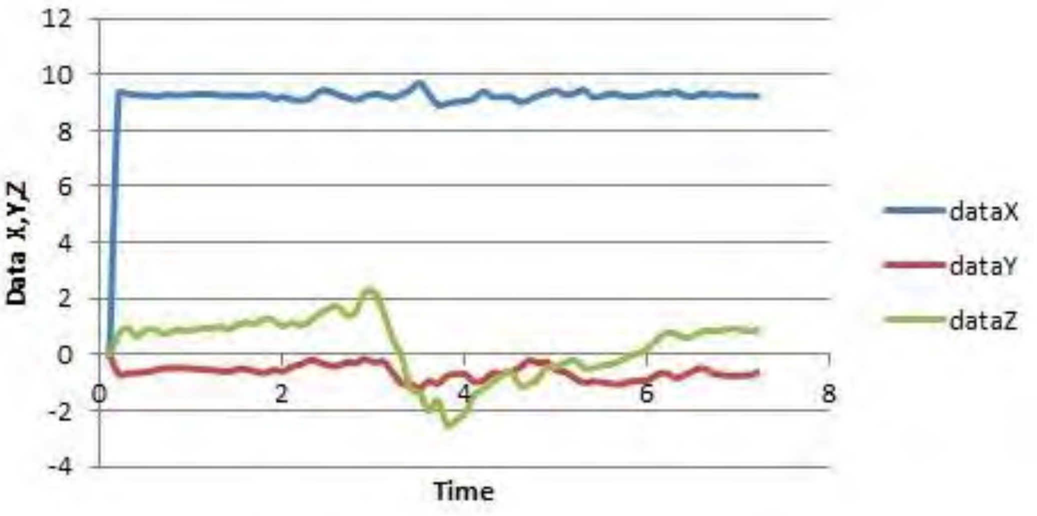


Figure 33. Backward movement acceleration on graph

The Z-axis data showed in Figure 33 a negative value spike and therefore signifies the backward movement of the device. For each negative value spike there can be sensed a backward movement. According to the movement reading from the Table3 the image will be enhanced with transitions for better experience which is showed in the Figure 34.

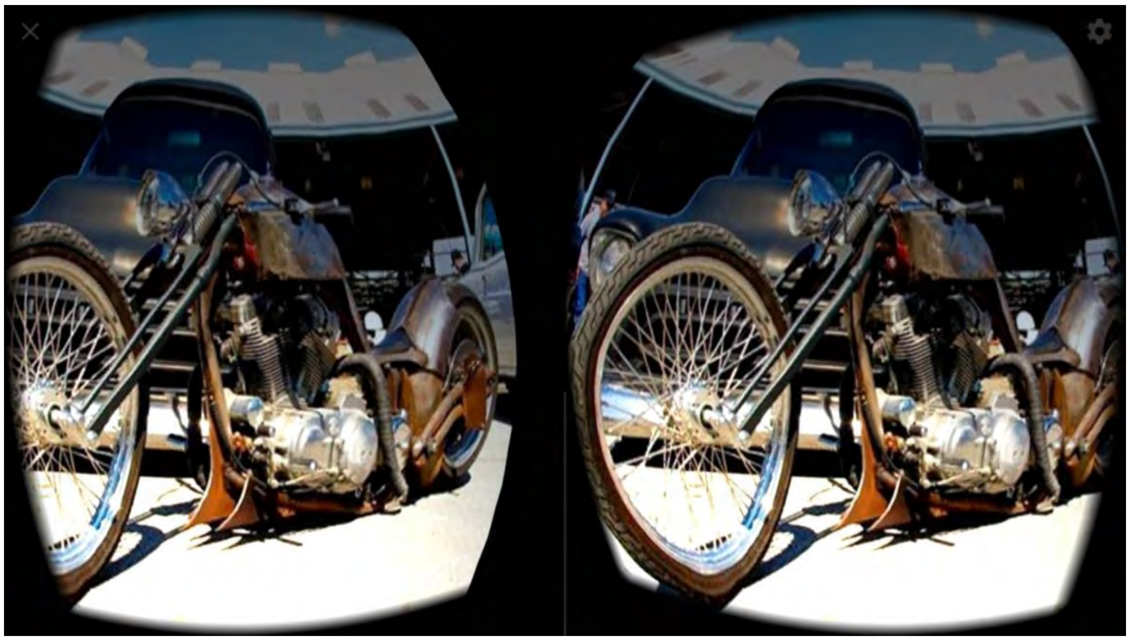


Figure 34. 2D Sample image (I) in VR view navigating backward

In Figure 35, while moving backward physically it shows that the object going further.

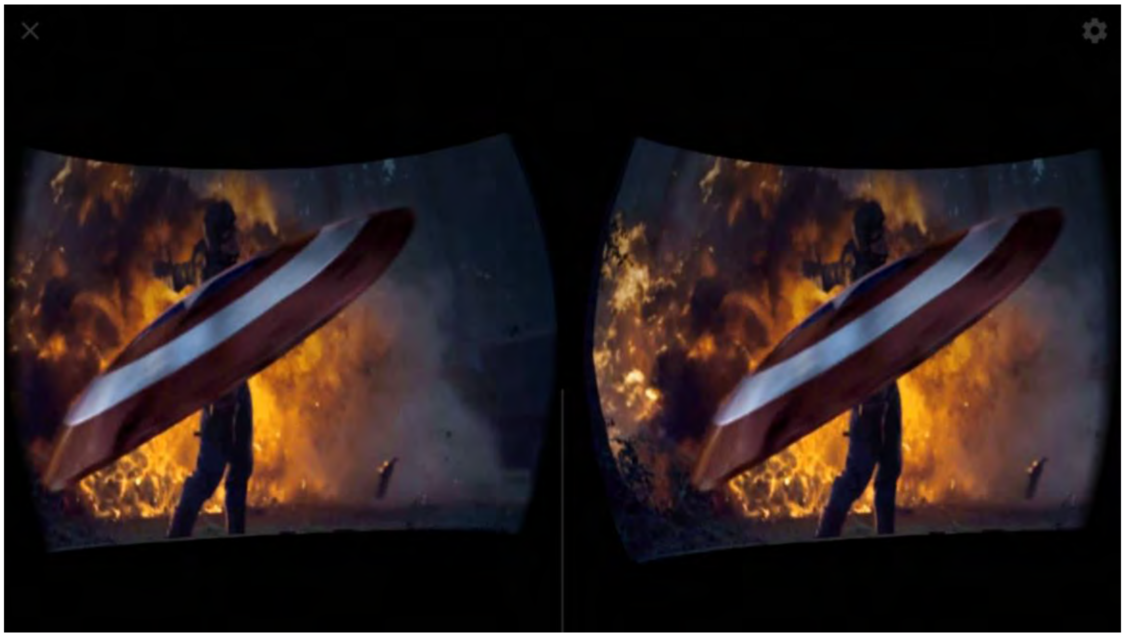


Figure 35. 2D Sample image (II) in VR view navigating backward

Similarly in the Figure 36, backward movement physically will give the sense of going backwards in the pseudo-space virtually.

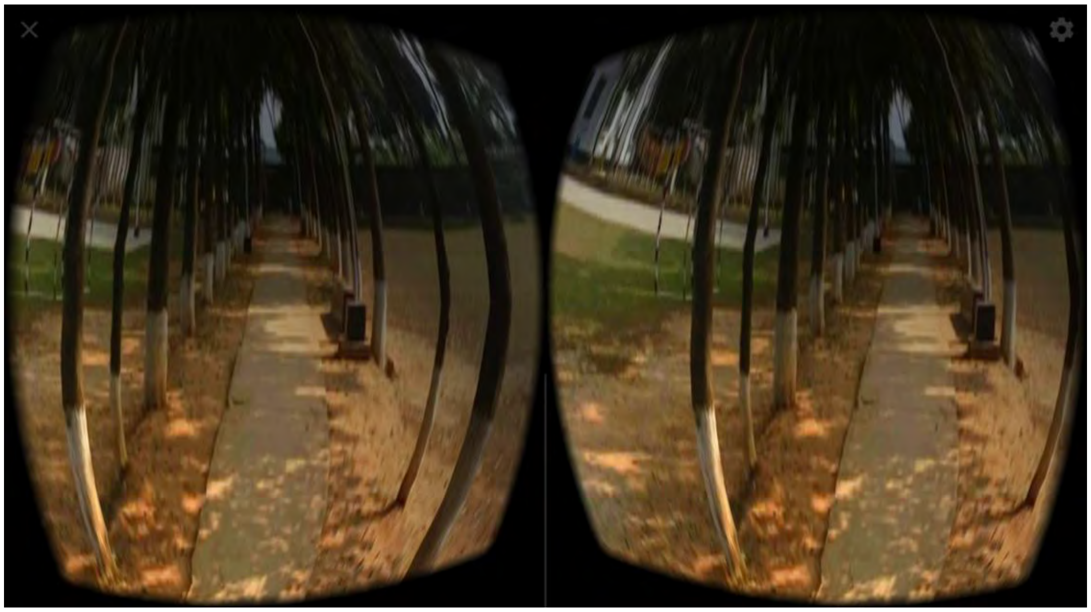


Figure 36. 360 Sample image in VR view navigating backward

## 4.2 Methodology for Accuracy

The proposed model performs five steps to integrate it with VR device. Firstly, the 2D image is segmented to two images, after that the depth map is calculated to generate the depth image. Thirdly the stereo pair of images is generated with the depth map and the original images. The stereo pair is then ported to the VR device with the Google cardboard simulation tools for VR device. The accuracy of the methodology is divided into two parts one is for the stereo pair and another one is for the VR integration.

For stereo pair, depth image generation and the segmentation of the image at the first step plays an important role. If after the segmentation of the image new created image is not as the ideal image it in length and pixel mapping then the image segmentation is not accurate. Without the accurate image segmentation, it is not possible to calculate the depth map of the image.

The computer generated depth map different from the depth map image that can be perceived from the sensors. As we know from equation from the depth calculation (12), the depth map depends on the Baseline T, f focal length in pixels and the disparity calculation. Thus there is a strong dependency of accuracy on baseline and disparity map.

The disparity error can be calculated through the following equation,

$$dD = \frac{dd * D\hat{}2}{f * T} \qquad (17)$$

Where, dD is depth error, dd is disparity error, D depth, f is focal length and T is baseline.

The disparity of error is 1.4um of normal VGA camera [39] for different focal length and base line of (T) the depth error can be calculated on the table 4.4 below

Table 4. Depth Error

| Disparity Error, dd(um) | Depth(m), D | Focal length, f(mm) | Baseline ,T (m) | Depth error, dD (%) |
|---|---|---|---|---|
| 1.4 | 10 | 14 | 0.024 | 9.7 |
| 1.4 | 15 | 17 | 0.033 | 3.75 |
| 1.4 | 11 | 23.5 | 0.029 | 3.25 |
| 1.4 | 19 | 24 | 0.026 | 5 |

From the above table 4.4 the depth error for the computer generated pair varies depending on the focal length and the baseline of our eyes.

Noise is found in the acceleration reading from the sensor manager. The noise can be handled by integration of the function data. But the accuracy falls exponentially as the data will show up to 20 m of error [23].

Therefore we did not reduce the noise and approximated the noise to be, (15)

$$\text{Noise} = A - g\,(\text{at rest}) \qquad (15)$$

Experimentally, $\text{Noise} \sim (+/-)\ 2\text{ms}^{-2}$

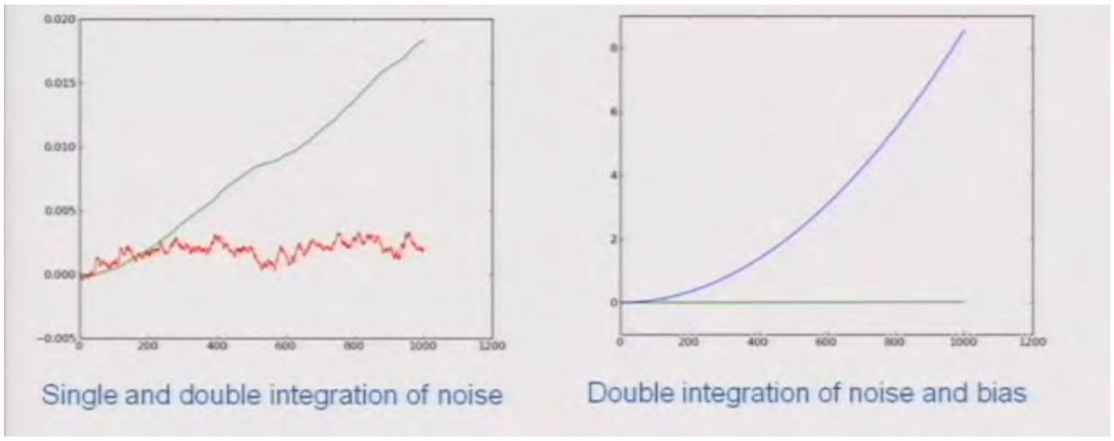


Figure 37. Double Integration to find linear movement [23]

From the Figure 37, it shows that the single and double integration of noise and their biasing. So, from this double integration is better cause it deviates more from which we can get better result.

## 4.3 Experimental Result Analysis

In our proposed model, the depth image is generated through computer and also the navigation part is also without the help of external sensors. If we compare the computer generated depth image model with the Kinect sensors depth image, the depth will be more diverse. The RGB-D sensors, such as Microsoft Kinect, are 3D sensors from which the depth image is created by calculating the distortion of the IR lights. [40]. But the sensors that are in use also has some limitation regarding the distance of the object from the sensor. The distance starts from millimeters to at max up to 4cm [41]. The standard deviation at different distances of plan to the sensors provides a similar type of quadratic curve in precision (Figure 38) [42]. The limited range for Microsoft SDK is above 800 millimeters for default range and go down to 500 millimeters for nearer range [43]. However, from

Figure 38 we can see that it provides measurement range starting at 500 millimeters until 4000 millimeters which is better range but after that it starts to deviate [42]. Whereas in the proposed model the depth image created via computer, for which the distance limitation is not find in our model. Though we face error for the focal length of the camera from which the 2D image is collected and also with the baseline by which the image is segmented.

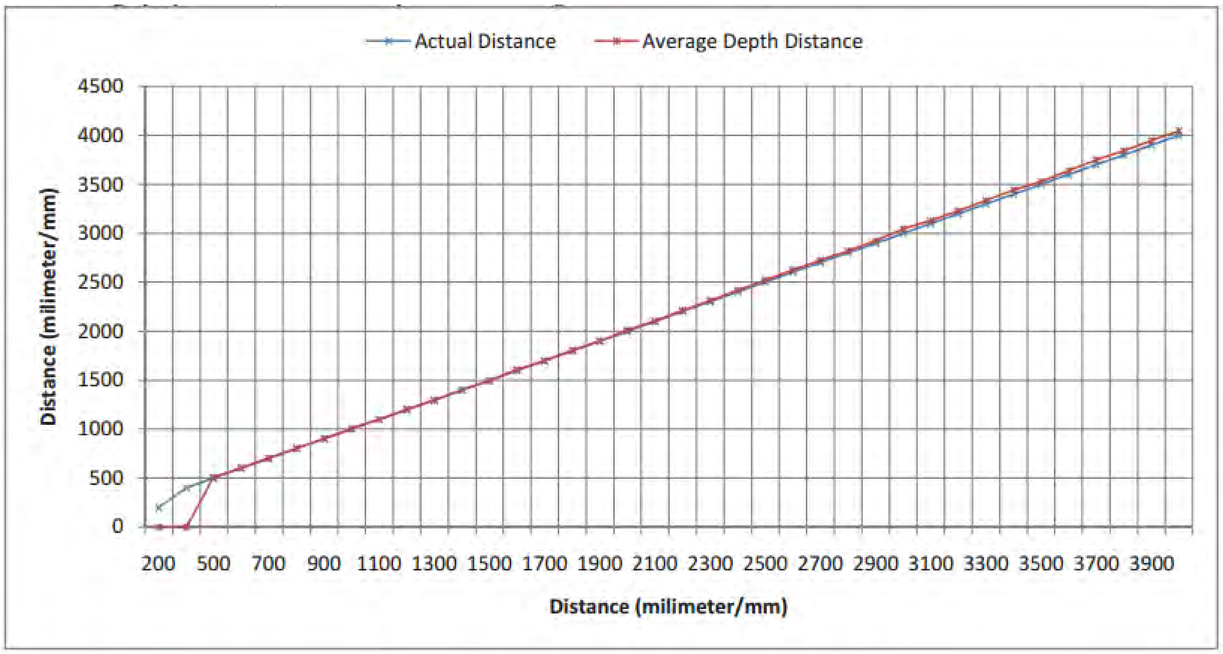


Figure 38. Comparisons between Average Depth Distance vs. Actual Depth Distance [42]

The negation system that we proposed are inexpensive than the available systems due to less use of sensors. Besides, using of the Google Card Board platform made it more cost effective [32]. The sensor uses long range IR sensors which is why the system becomes expensive. Again, the existing system reads position therefore it takes three dimensional movement reading. But in our system it, takes reading only in one axis as they we are working in 2D image. All in all, our proposed model roughly gives a better experience for 2D image which is converted for 3D visualization.

Chapter 05

# Conclusion

## 5.1 Concluding Remarks

To sum up, we have demonstrated a technique in which one can experience a 2-dimentional image as if it is 3-dimentional and will be allowed to navigate through mobile sensors to certain extend. Since, this has been possible only with external sensors; our model offers newness and originality. Our proposed methods can be applied to build applications, games and can open up unexplored grounds on mobile sensors and image motion processing. The technique of getting depth information from single image is also noteworthy. Not only it has virtual reality applications but also can be used to perfect Mixed Reality to some extent.

## 5.2 Limitations

In our proposed model the creation of depth image is based on taking information from a single image which is a re-modification of previous models. Since, it is completely Computer Generated rather than using Kinect or dual camera for depth sense, the results will not be as accurate it might in receive through the sensors. In table 4.4 the error rate varies depending on the disparity error which varies camera to camera. That is why the results are close enough to the original result rather than being accurate as sensor based depth image. The accelerometer information during navigation is not the cleanest, since noise and gravity is always acting during taking reading from the sensor. Adding up as well, the physical navigation of the wearer of head-mount display cannot exceed 2-3 meters as it will be hazardous for the wearer.

## 5.3 Future Works

Creating computer generated stereo pair can be improved more by using high GPU processing. Besides, multiple angle of 2D images can give the depth image more accurate which can be computed through GPU processing. This will generate more accurate image much alike Kinect sensors. Since Virtual Reality simulations are much preferred nowadays, it can solve many real life problems regarding investigations, education, entertainment etc. If the depth navigation is accessed to even further extend, Crime scene investigations, 3D simulation from 2D images, VR 3D interactive movies etc. can be some ground breaking applications of the technology.

# References

[1] C. Boletsis, "The New Era of Virtual Reality Locomotion: A Systematic Literature Review of Techniques and a Proposed Typology", *Multimodal Technologies and Interaction*, vol. 1, no. 4, p. 24, 2017.

[2]A. Steed and S. Julier, "Design and implementation of an immersive virtual reality system based on a smartphone platform", *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, 2013.]

[3]S. LaValle, A. Yershova, M. Katsev and M. Antonov, "Head tracking for the Oculus Rift", *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[4]A. Sabale and Y. Vaidya, "Accuracy measurement of depth using Kinect sensor", *2016 Conference on Advances in Signal Processing (CASP)*, 2016.

[5]W. Rollmann, "Zwei neue stereoskopische Methoden", *Annalen der Physik und Chemie*, vol. 166, no. 9, pp. 186-187, 1853.

[6] Gemshein, H., & Gemshein, A. (1969). The History of Photography from the Camera Obscura to the Beginning of the Modern Era. NY, McGraw-Hill

[7] W. Zuiderbaan, B. Harvey and S. Dumoulin, "Image identification from brain activity using the population receptive field model", *PLOS ONE*, vol. 12, no. 9, p. e0183295, 2017.

[8]S. Izadi, A. Davison, A. Fitzgibbon, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges and D. Freeman, "KinectFusion", *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, 2011.

[9] Makhzani Niloufar, Kok-Why Ng, and Babaei Mahdi, “DEPTH-BASED 3D ANAGLYPH IMAGE MODELING,” *International Journal of Scientific Knowledge*, vol. 4, Mar. 2014.

[10] J. Ko, M. Kim, and C. Kim, “2D-to-3D stereoscopic conversion: depth-map estimation in a 2D single-view image,” *Applications of Digital Image Processing XXX*, 2007.

[11] Z. Ganatra and R. Chavda, *International Journal of Innovations & Advancement in Computer Science*. 2014.

[12]Ke Lu, Xiangyang Wang, Zhi Wang and Lei Wang, "Binocular stereo vision based on OpenCV", *IET International Conference on Smart and Sustainable City (ICSSC 2011)*, 2011.

[13]Arts.rpi.edu,2018.[Online].Available:http://www.arts.rpi.edu/~ruiz/stereo_history/Howtomake3D.htm. [Accessed: 19- Jul- 2018].

[14]Z. Lu, S. Rehman, M. Khan and H. Li, "Anaglyph 3D Stereoscopic Visualization of 2D Video Based on Fundamental Matrix", *2013 International Conference on Virtual Reality and Visualization*, 2013.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2003.

[16] M. Brown and D.G. Lowe. Automatic panoramic image stitching using invariant features. Int. J. Computer. Vision, 74(1):59–73, 2007.

[17]"The Teardown: Oculus Rift 3D gaming head-mounted display", Engineering & Technology, vol. 8, no. 5, pp. 88-89, 2013.

[18]A. Ng, L. Chan and H. Lau, "A low-cost lighthouse-based virtual reality head tracking system", *2017 International Conference on 3D Immersion (IC3D)*, 2017.

[19]P. Wozniak, O. Vauderwange, A. Mandal, N. Javahiraly and D. Curticapean, "Possible applications of the LEAP motion controller for more interactive simulated experiments in augmented or virtual reality", *Optics Education and Outreach IV*, 2016.

[20]K. Arai, H. Tolle and A. Serita, "Mobile Devices Based 3D Image Display Depending on User's Actions and Movements", International Journal of Advanced Research in Artificial Intelligence, vol. 2, no. 6, 2013.

[21]K. Fujinami, "On-Body Smartphone Localization with an Accelerometer", Information, vol. 7, no. 2, p. 21, 2016.

[22]"Motion sensors", Android Developers, 2018. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion. [Accessed: 20- Jul- 2018].

[23]D. Sachs, "Sensor Fusion on Android Devices: A Revolution in Motion Processing", 2010, Available: http://youtu.be/C7JQ7Rpwn2k . [Accessed: 20- Jul- 2018].

[24] G. Llorach, A. Evans, J. Agenjo and J. Blat, "Position estimation with a low-cost inertial measurement unit", *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, 2014.

[25]"Tips for setting up the base stations", Vive.com, 2018. [Online]. Available: https://www.vive.com/eu/support/vive/category_howto/tips-for-setting-up-the-base-stations.html. [Accessed: 02- Aug- 2018].

[26]Depth map from stereo pair", *Forums.ni.com*, 2018. [Online]. Available: https://forums.ni.com/t5/Machine-Vision/Depth-map-from-stereo-pair/td-p/1488176. [Accessed: 01- Aug- 2018].

[27]A. Pradhan, A. Kr. Singh and S. Singh, "An Approach to Calculate Depth of an Object in a 2-D Image and Map it into 3-D Space", *International Journal of Computer Applications*, vol. 119, no. 15, pp. 27-32, 2015.

[28]"Semi-Global Block-Matching Algorithm - NI Vision 2012 Concepts Help - National Instruments", Zone.ni.com, 2018. [Online]. Available: http://zone.ni.com/reference/en-XX/help/372916M-01/nivisionconceptsdita/guid-53310181-e4af-4093-bba1-f80b8c5da2f4/. [Accessed: 20- Jul- 2018].

[29] Battiato, S.; Capra, A.; Curti, S.; and La Cascia, M, "3D Stereoscopic Image Pairs by Depth-Map Generation", in Proc.of 2nd International Symposium on 3D Data Processing Visualization and Transmission, 3DPVT ,2004

[30]"Cardboard Camera Converter", Storage.googleapis.com, 2018. [Online]. Available: https://storage.googleapis.com/cardboard-camera-converter/index.html. [Accessed: 20- Jul-2018].

[31]"VrPanoramaView", Google Developers, 2018. [Online]. Available: https://developers.google.com/vr/reference/android/com/google/vr/sdk/widgets/pano/VrPanoramaView. [Accessed: 20- Jul- 2018].

[32]W. Powell, V. Powell, P. Brown, M. Cook and J. Uddin, "Getting around in google cardboard – exploring navigation preferences with low-cost mobile VR", *2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR)*, 2016.

[33]"GVRPanoramaView Class", Google Developers, 2018. [Online]. Available: https://developers.google.com/vr/ios/reference/interface_g_v_r_panorama_view. [Accessed: 20-Jul- 2018].

[34]U. Technologies, "Unity - Manual: Shader Reference", Docs.unity3d.com, 2018. [Online]. Available: https://docs.unity3d.com/560/Documentation/Manual/SL-Reference.html. [Accessed: 20- Jul- 2018].

[35]"SensorManager | Android Developers", *Android Developers*, 2018. [Online]. Available: https://developer.android.com/reference/android/hardware/SensorManager. [Accessed: 02- Aug-2018.

[36]"Accelerometer Input - Unity", Unity, 2018. [Online]. Available: https://unity3d.com/learn/tutorials/topics/mobile-touch/accelerometer-input. [Accessed: 20- Jul- 2018].

[37]"Stereoimage",En.wikipedia.org,2018.[Online].Available:https://en.wikipedia.org/wiki/Stereo_image. [Accessed: 01- Aug- 2018].

[38]"Captain America: The First Avenger Blu-ray", *Blu-ray.com*, 2018. [Online]. Available: http://www.blu-ray.com/movies/Captain-America-The-First-Avenger-Blu-ray/22069/. [Accessed: 01- Aug- 2018].

[39]F. Xiao, J. Farrell, P. Catrysse and B. Wandell, "Mobile Imaging: the big challenge of the small pixel", *Digital Photography V*, 2009.

[40]Alhwarin F., Ferrein A., Scholl I. (2014) IR Stereo Kinect: Improving Depth Images by Combining Structured Light with IR Stereo. In: Pham DN., Park SB. (eds) PRICAI 2014: Trends in Artificial Intelligence. PRICAI 2014. Lecture Notes in Computer Science, vol 8862. Springer, Cham

[41]K. Khoshelham and S. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", *Sensors*, vol. 12, no. 2, pp. 1437-1454, 2012.

[42]L. Chin, S. Basah, S. Yaacob, M. Din and Y. Juan, "Accuracy and reliability of optimum distance for high performance Kinect Sensor", *2015 2nd International Conference on Biomedical Engineering (ICoBE)*, 2015.

[43]T. G. and P. A. M.R. Andersen, T. Jensen, P. Lisouski, A.K. Mortensen, M.K. Hansen, "KINECT DEPTH SENSOR EVALUATION FOR COMPUTER VISION APPLICATIONS," 2012