Novel Gaming Experiences with Advanced Illumination through Augmented Reality: Interim Report

Terence Tse
Supervisor: Benjamin Glocker
Second Marker: Abhijeet Ghosh

May 11, 2015

Contents

Abstract

Oh

Introduction

I want to create the world.

Augmented Reality(AR) has come into the limelight once again with the recent release of the Google Glass. AR has been around for a while now, most of the time being thought of the future of technology but with applications ending up with limited use, less than satisfying performance and being gimmicky, it has been hanging around in the background for a while.

AR in a nutshell is reality, augmented! A view of the real world is presented to the user but it has been enriched in some sort of way. If you are unaware of what looking through a Google Glass is like, one pretty well known example is the film Terminator. When seeing from Terminator's viewpoint, we are presented with a red screen, this is what a normal person would see but in red. However, on the screen, it has a crosshair, which lock onto a person, their statistics in the right hand corner, when it's your target, the words flash in view. This is what augmented reality is.

Of course, we're not all cyborg assassins, hence why things like the Google Glass have been created. Our eyes already give us a lot of information about the world, but having even more information presented to us than what we naturally have is Augmented Reality. Already, anyone with a semi-decent mobile phone can play around and take benefits from Augmented Reality, it does not take large amounts of processing power to produce an AR image. The key is that AR is real time, our view gets updated as we see it.

Augmented Reality is not just limited to some head mounted device. In addition to our phones, we can use tablets or just plain computers in which to play with AR. All we need is a camera which can capture the world and then allow us to view it either with additional information or even diminished information! One use case is in the British Maritime museum, where visitors can aim cameras at special AR exhibits and have the item explained to them visually. The range applications of AR are overwhelming, imagine a Dinosaur coming to life in a History museum and you can see it navigating the halls. Imagine having an AR mirror that superimposes clothes onto your body that fit, without having to go and search and try on the clothes themselves. Imagine being on a construction project and you are able to view the outcome on your tablet, even if the old structure has not been cleared away for the new project to begin! Augmented Reality can do all of that already and the scope of new applications is, no pun intended, as far as the eye can see!

As previously stated, AR has many applications in terms of games and entertainment. These come in all different styles, from simple use of a camera, to using fixed feature points as markers, or using patterns as feature points to identify placement of objects in the scene. Right now, these feature points are either very simple or immutable. They are something that the computer can directly identify which is obviously good for performance reasons. However, what about the ability to create your own feature points on the fly? If we able to immediately draw feature points and by extraction of these points, augment reality based off of these points, it would greatly improve the experience of Augmented Reality, removing the need to create the marker on a computer and just, for example, grabbing a piece of paper, drawing it yourself and have it work straight away! I shall refer to this as a feature drawing throughout the rest of the report.

However, AR may also be used for more commercial and other practical uses. When people think of AR, it usually defaults to thinking about games and indeed there is a lot of future in that industry with AR, there are also Educational uses, Project and Landscape Planning, Commerce and Animation, the list goes on. The problem right now is that simple AR applications which augment a scene with, say a dinosaur or some other object, those generated images are still quite obviously generated by a computer, especially when they try to get more complex. In a bright room, you may place furniture inside which looks semi-realistic, it has a computed shadow and gradient change. However, an ideal situation is when we augment reality and it doesn't even feel as if anything is different. In other applications, the objects may not even have an advanced lighting model, solid colours with no depth or obscurance may be modelled, thus not even a shadow is created making it extremely obvious (whether this is desired or not) to us which objects in the scene have been artificially generated. It is harder to appreciate this as an extension of reality when it is so obvious that this object does not belong in the scene.

This project is focused around these points. First, I'd like to make additions to the AR realm; I will focus around using Augmented Reality to provide not only a novel gaming experience, but also one that can be transferred over to more practical use cases while appearing realistic in addition to have users generate a set of feature points manually in the real scene. One way that we can drastically improve the realism of a computer generated image is to improve the illumination model to such a degree that the object looks like it belongs in the view.

Project Goals

The bigger picture and where this project hopes it can contribute and make headway into is permitting a dynamic and responsive AR application that allows creators to quickly draw

up environments which can then be visualised. Environments can also be changed in real time and will be reflected in the AR without need for additional user interaction. The completion of this task will have numerous transferable benefits that were mentioned in the SECTION ABOVE.
However, the amount of work that this goal encompasses coul be infintissimal depending on what kind of features would be deemed necessary, good or just an extension. There are also numerous parts of a project like this could be analysed in much further detail. As a result, the objectives for this project are:

- Create a functioning Augmented Reality program which takes a video stream and overlays a 3D landscape on a contour map created by the user. The contour map may also have other feature points which indicate other 3D structures to be rendered.
- Using a method of video and scene tracking/recognition, allow users to make changes to their contour map and have the landscape update in real time. This essentially allows dynamically changing Augmented Reality landscapes.
- Perform an analysis on the performance and responsiveness of the final product of this project. This will be done through user testing.

Background

Augmented Reality

What is Augmented Reality(AR) and why should we be interested in it?

In recent years, it becomes harder and harder to concisely describe what constitutes as Augmented Reality. The reason for this being that as technology improves more and more things are being achieve that allow users to extend their perceived reality with extra data.

Simply put, Augmented Reality is when the environment around you can be extneded by combining both real and virtual data components. Azuma's paper[3], *A Survey of Augmented Reality*, describes AR as having the following three characteristics:

1. Combines Real and Virtual (data)
2. Interactive in real time
3. Registered in 3D

While these are the basic things that constitute AR, there are three ways, outlined in Mackay's paper[8], *Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers*, in which you can augment reality.
These are:

1. Augmenting the User
2. Augmenting Physical devices/objects
3. Augment the Environment

<u>Augment the User</u>
Augmenting the user is having the user carrying, wearing or using a device that provide the user with extra information. When we talk about augmenting the user, the devices we talk about are usually heads-up displays (HUDS) which the user will have a screen in front of their eyes in some shape or form. One example of this is Google Glass, allows users to project information about what they see or their GPS location onto parts of the glass. If the user was looking at the Eiffel Tower through their glass, with GPS coordinates and a scan of the shape of the Eiffel Tower, Google Glass would be able to tell that what the user was looking at. From there, it can project information about the landmark to the user which would not have been accessible so easily otherwise!

This is one such trivial application of user worn AR, it is very useful for lesser known things, for example, instead of looking at a famous landmark, the user could be an engineer, looking at a complex piece of machinery and a HUD allowing them to get information on each component. This is also being trialled in Medicine, where doctors can use Augmented Reality to help teach surgical procedures or even help them in analysis of medical images such as X-Rays.

Augment Devices

When an object or device has a small computer placed into it, it can be considered as an augmented device. One example of this is again in surgery and medical environments. Having tools such as scalpels or other medical equipment fitted with computers to plot out how much of the skin has to be cut, for example, allows the user to receive much more information than they would be able to than without. Mackay also talks about these briefly in her paper.

Augment the Environment

This form or AR doesn't add any additional hardware to devices nor the user. It focuses on having external devices, such as projectors or cameras take input from the user and use that to transofrm the environment that they are in or are manipulating. This class of AR is much wider than the two aformentionned as there are less things imposed on an object/person which allow a larger spectrum of devices to be used.

An example of Augmenting the environment is an AR keyboard, a projector displays a keyboard in front of where it is placed, the user can then "press" keys and the kyboard will be able to detect which key has been selected and translate it into computer input.

A more common device is your smart phone! Tablet or even computer screens can be included in this form of AR. There are plenty of mobile applications that allow users to experience AR in their own unique way. An example used in the British Maritime Museum is the ability to take a tablet with predownloaded apps that allow users to engage in an AR game. Through the game, which is primarily aimed at children, allows them to learn about the exhibits in a more fun and interactive way. There are also similar applications in retail, for example in retail! Augmented reality mirrors are being used to avoid the time in the changing room and allow customers to "try" on pieces of clothing through AR by overlaying the garment over their captured video.

Another fun implementation of AR is the Augmented Sandbox. A projector and a camera are used to transform a sandbox into a remoldable landscape. The camera uses depth sensing and the input from here is transformed into lanscape height and the projector correspondingly projects land or water onto these areas. The magic happens when the sand is manipulated and the whole system re-renders the landscape in real time to provide an interactive augmented environment! Sarah Reed from the University of California lead the authoring of the persentation about this piece of tehnology[12].

Computer Vision and Image Processing

A key part of this project is the Computer Vision aspect. The field of Computer Vision involves the understanding of data from images and pictures. "Understanding" is a broad term that encompasses many ways of manipulating or processing images in order to extract features or points of interest that can be then used by the computer.

Edge Detection

In an image, an edge if a set of pixels that separates to disjoint regions.

A key part of Computer Vision is line and edge detection. Edge detection is the very beginning of how we can start to identify different objects within an image. In addition, by stripping an image down to its comprising edges, we greatly decrease the amount of information in the image whilst preserving the important parts which we want to operate on. As a result, how edges are detected have a large contribution to what we can do with certain images.
Edge detection tries to find the set of pixels that separate two disjoint areas of an image by analysing sets of neighbouring pixels and looking for discontinuities in intensity or texture. A variety of factors can cause these (local) discontinuities:

1. Object colour and texture. If the object has any change in colour or texture, then where the change occurs, an edge will be detected.

2. A change of surface normal. In the real world, we have 3D objects, as a result, there will be a surface normal change when we consider, say, the top face of a box against the side face. There is an edge between the two and this is partly also due to the next factor.

3. Illumination. A change of illumination or simply due to some part of the obejct in question casting a shadow will cause some regions of the image to be darker or lighter in intensity, even if said regions are part of the same object. Edges will be detected in this instance.

4. Two objects. This a large part of why we even do edge detection, to detect objects. If there are two objects, then, likely, there will be a difference in colour and intensities

of their constituant pixels. However, this is not always the case and also as can be reasoned from above, these regions may belong to the same multi-coloured object.

When it comes to
We can distinguish edge detectors into two sets, the Laplacian of Gaussain detectors and the Gradient-based detectors.

Gradient-based detectors

Gradient edge detectors look for changes of gradient (first-order) of neighbouring pixels. Some common gradient-based detectors are Sobel, Prewitt, Canny and Robert. Each of these detectors starts off with an image and an x and y kernel. The kernels help determine the gradient change and y directions of the image, different kernels (size and values) are what separate the different Gaussian-based detectors. The final gradient magnitude for a given pixel is then the square root of the sum of these:

$$G = \sqrt{\Delta x^2 + \Delta y^2} \tag{1}$$

Users can then set a threshold value for which any pixel with a final gradient that is above that threshold will be treated and marked as an edge, the others will be discarded.

Laplacian (of Gaussian)-based detectors

Laplacian based detectors are similar to Gradient-based ones except that instead of taking the first derivative, they consider the second derivative. Their goal is to look for zero-crossings, i.e. local maxima in gradient changes (i.e. looking for when gradient starts going negative). The change of gradient sign, which occurs when second order derivative is 0, indicates that the intensity of the observed pixels begins changing also. This method can be coupled with a Gaussian kernel which serves to de-noise the image by smoothing out the pixel intensities.

Canny Edge Detector

There have been numerous methods of detecting edges put forward, some Gradient-based and Laplacian based edge detection methods are outlined in *A Comparison of various Edge Detection Techniques used in Image Processing* [14]. However, the most popular edge detector out there is the Canny Edge detector, put forward by John Canny in his paper *A Computational Appraoch to Edge Detection*[5]. The paper dicusses a methematical approach to edge and ridge detection which is then analysed on in Ding and Goshtasby's paper, *On the Canny Edge Detector*[6]. Rashmi's paper, *Algorithm and Technique on various Edge Detection: A Survey* [11] compares various edge detection algorithms and finds that

the Canny edge detector comes out on top, with Maini's paper *Study and Comparison of Various Image Edge Detection Techniques* [9] showing the same result.
The Canny Edge detector falls within the Gaussian-based Edge detectors and can be described algorithmically below:

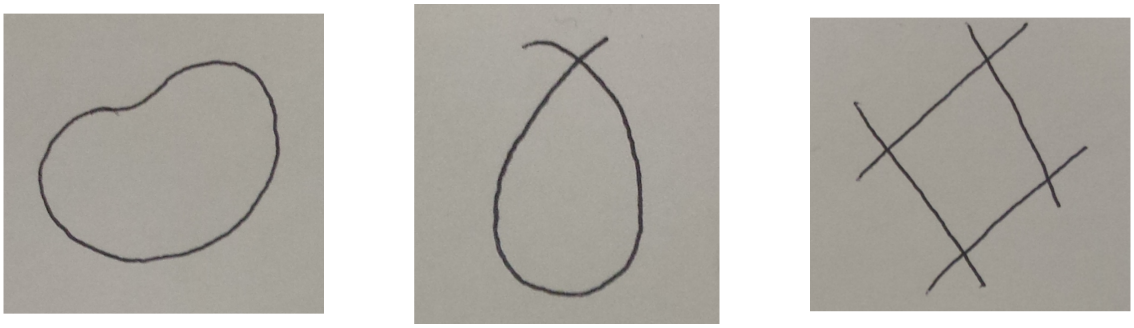
- Apply a Gaussian filter over the image to remove noise by smoothing
- Take the Gradient of the image pixels over each direction, just as would be done in a normal Gaussian-based detector.
- Apply non-maximum suppression such that only local maxima are marked as edges (the pixel has a greater gradient magnitude than all of its neighbours) resulting in a thin edge. The thinning of found edges is to remove some affect of the blur. After this, perform double thresholding, those higher than the threshold are "strong" edges and those under are weak edges.
- Another round of removing edges is applied by using hysteresis. This involves identifying "strong" edges,and then removing all edges that are not connected directly to these strong edges.

The steps are more fully explained in the paper by 09gr820, *Canny Edge Detection*[1].

Open and Closed contours

We understand a closed contour to be a loop. By this we mean that if we had to draw a closed contour, we begin at a point on a piece of paper and end at that point, either by not removing our pen from the page or, if done so, will resume at the same point to finish up the contour. The existence of white space between points on a contour mean that the contour is open, thus not closed. Examples of open and closed contours can be seen in Figure 1 and Figure 2.

For feature detection, and especially within this project, we look for *closed* contours, they provide us with the most information. An open contour may be linkeed to some kind of feature point but if it is not closed, its usefulness is very low. In particular, we want closed contours so that any contours that are drawn within them can then lend themselves to a natural hierarchy/ordering. In the context of this project, those within closed contours will represent areas that are of higher elevation than the area contained between itself and the outer contour. When contours are not closed, it becomes harder for us to identify this hierarchy and ordering within the image, which would mean a lot more difficult computation to be done to achieve our goal of generating a corresponding landscape. Indeed, this is the problem of many similar projects.

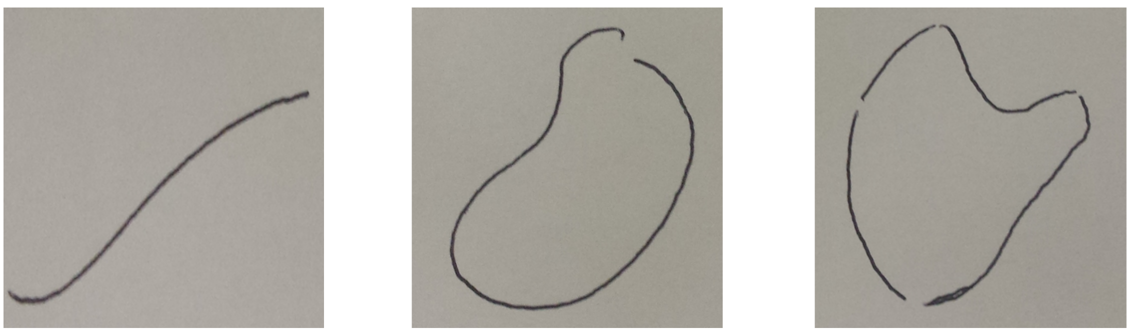


(a) Contour is fully closed, no gaps

(b) Contour is closed, but with artefacts on the ends

(c) This is also a closed contour

Figure 1: Examples of **closed** contours



(a) A line/edge counts as an open contour

(b) Contour open as beginning doesn't connect to end

(c) This contour is open with several gaps

Figure 2: Examples of **open** contours

Mathematical Morphology

One technique to help close contours/loops in Computer Vision is *Closing*. To understand *Closing*, we must first introduce the ideas of Mathematical Morphology (MM). MM is a method of analysing geometric objects and structures, typically used on images but can also be applied to other things such as graphs. Within MM, there are four basic operators, but for the purpose of this project, I shall only go into detail about the first three:

- Erosion
- Dilation
- Closing (Dilation followed by Erosion)
- Opening (Erosion followed by Dilation)

The operators were originally defined for binary images and so shall be described below in that sense, however, the functionality has been extended to include grayscale images as well, which is very useful when dealing with photographed images or video feed as will be done within this project.
For these operators, let us assume there is a binary image, with pixels holding values of either 0 or 1. In correspondence to a typical colour image, 1 indicates white, a brighter area, of the image and 0 corresponds to a black pixel. In the erosion and dilation operators, we have this binary image as input. We also have a corresponding kernel which we will convolute with the image. This kernel, sometimes referred to as a structuring element, usually takes the shape of a circle, a 3x3 square or a cross. To achieve different results from these operators the kernels can be changed, for example to a 10x10 square hich will cause a more drastic shrink or growth of bright areas. For the purpose of explanation, I will assume we are using a 3x3 kernel.

Erosion

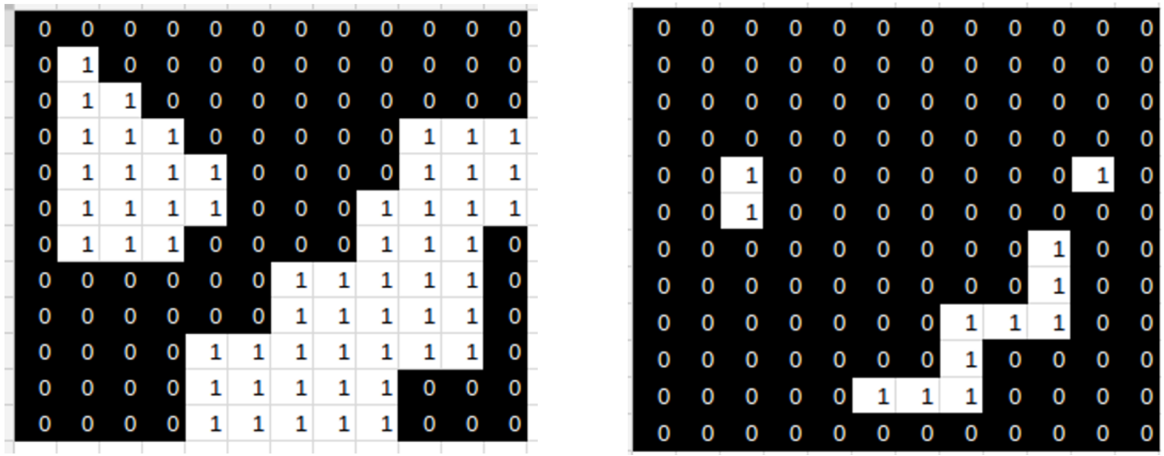
Erosion causes bright areas of images to shrink.

Having the 3x3 kernel anchored at its centre square/pixel, we place align the kernel with every single pixel on the binary image by superimposing the kernel's anchor over it. Within the other kernel pixels (a 3x3 neighbourhood of the pixel in question), we set the pixel under the anchor to the minimum value in this neighbourhood. Thus, the only way a pixel can remain white in a binary image is if it starts off as white and all of its neighbours with regard to the kernel shape, are white too. If this is not the case and even 1 is 0 in its neighbourhood, the pixel considered is set to 0. The effect of this can be seen in Figure **??**. As you can see from the result, the original shapes have become much thinner and there are now 3 shapes instead of 2. When combined with dilation, it is clear how some contours can be "opened" as erosion comes as the last step. The result shown in Figure **??** shows how what could be the constituent pixels of a contour being broken down into 2 separate objects of interest, thus *opening* the contour. In addition, erosion can be used to remove small, likely unimportant, parts of images, such as noise, since any noise that isn't large enough to be a problem, would be fizzled out by the kernel, whose size can be changed to affect noise sensitivity.

Dilation

Dilation causes bright areas of image to expand.

Dilation performs the opposite of what erosion would do. We still perform convolution with our 3x3 kernel but instead we set the pixel below the kernel anchor to be the *maximum*



(a) Original Binary Image

(b) Eroded Image

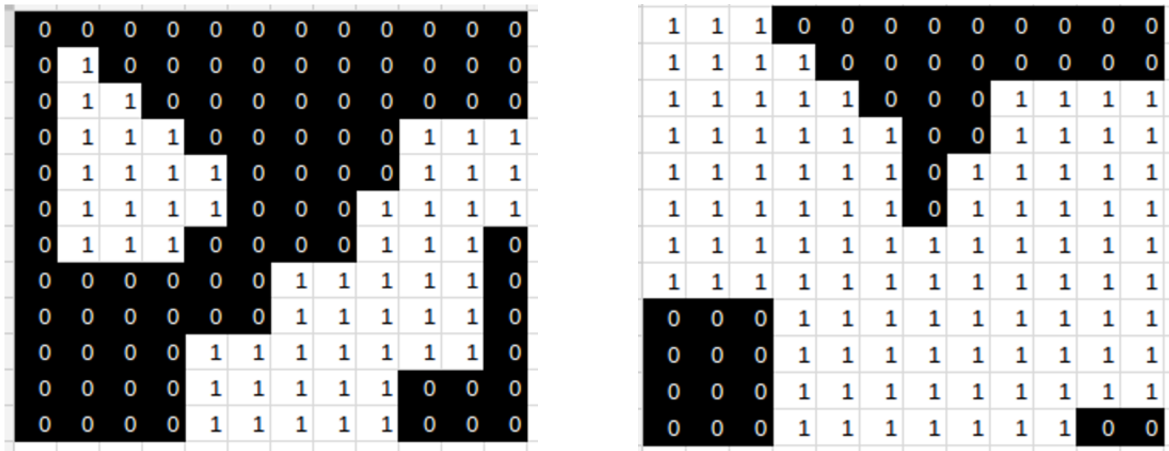
Figure 3: Erosion

value within the 3x3 neighbourhood. As a result, if there is even a single white pixel in the neighbourhood, the current pixel will turn white. The only case this will not happen is if the pixel started off with a pixel value of 0 and all of its neighbours also had a value of 0. Figure 4 shows the result of dilation with the same original image as shown in the Erosion section above.
It can be observed from the result above that a dilation greatly increases the size of the bright area. In addition, if bright areas are suitably close to one another, it can cause the areas to join into one through the dilation. This is how contours with small gaps are able to be closed and will aid in the contour detection process. Again, the kernel size and shape may be changed to cover a larger area (thus join bright areas further apart) or otherwise depending on how sensitive you want the operation to be.

Closing

Closing is Dilation followed by Erosion.

Closing is very similar to dilation but it aims to preserve more information. The Closing operation is no different to erosion or dilation in that we provide a kernel with which to onvolute the given image with. To perform Closing, the image first has a dilation operation applied to it with the given kernel. After this, erosion is done with the same kernel on this dilated image.
By just performing dilation, we do manage to close small gaps in the image, however, every pixel is equally distorted by the kernel and so the resulting image will look quite different to the original shapes, as can be seen in Figure 4 above. In order to reduce as much of this



(a) Original Binary Image

(b) Dilated Image

Figure 4: Dilation

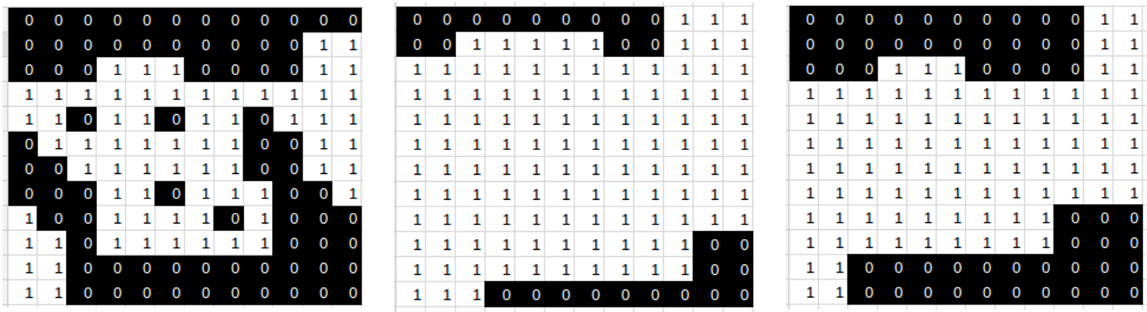
distortion as possible, erosion is used. The overall affect of this is to make the resulting image be as similar as possible to the outline of the provided original image. Closing is idempotent, after one pass the resulting image will not change from further Closings with the same kernel. The effect of closing is illustrated in Figure 5.
The original images starts off with some holes within the image. If we take this to be a contour discovered by an edge detector, then we can show how we can close it with *Closing*. After the Dilation stage, the small holes within the original image have been closed, in addition, the boundaries of the bright parts of the picture have been joined due to their proximity and the kernel shape. However, as seen in Figure 5 (b), the shape has changed a little from the original picture. After applying eroision, seen in Figure 5 (c), the final result is much closer in shape to the original picture, except it has its holes filled and the bright areas have joined. This would have closed the contour, if it were one, in one of the best ways possible, preserving most of the original shape and not losing too much fine detail through enlarging the bright area!

Image Registration

Feature Extraction

Image Differencing

For users to be able to draw their own landscapes, there has to be some way of recognising when there has been a change, physically, on paper. To automate this process will enable the program to decide for itself when there has been an update to a drawing, rather than



(a) Original Picture (b) Dilation stage (c) Erosion stage

Figure 5: Closing operator step by step

a user having to confirm each time there is an update. Image differencing is the primary way of achieving this kind of detection. There are many different ways of actually working out whether there is a "difference" between two images and some will be further explained below. The general method for image differencing from a video feed is taking two frames, at time t_s and a time in the future t_{s+c} and compare the two.

Simple Intensity Differencing

The simplest form of finding the difference between two images is comparing their pixel intensities.

$$
\begin{aligned}
A &\text{ is Image at time } t_s \\
B &\text{ is Image at time } t_{s+c} \\
D &\text{ is the difference Image} \\
I(pixel) &\text{ is the Intensity of the supplied pixel} \\
\\
D_{i,j} &= |I(A_{i,j}) - I(B_{i,j})|
\end{aligned} \tag{2}
$$

Once the Difference Image has been computed, you can then perform a sum over all the pixels to get an indication into how much "change" there was. Depending on if the image is grayscale, binary or RGB will give different ranges of numbers, the higher the range, the less this final sum will be able to tell you about the change in the image.
A popular way to deal with this problem is to perform thresholding on the Difference Image and is explained in Rosin's Paper *Thresholding for Change Detection*[13]. Again, there are different type of thresholding such as Truncation, to-Zero, Binary and all of their inversions. For the purposes of this project, we will only look at Binary thresholding.

Binary thresholding takes a threshold value, τ, which is acquired either through statistical or empirical methods. With the threshold, any pixel value in the Difference Image that is below the threshold is effectively ignored and set to 0. The pixels above the threshold preserve their values. This method is very quick, especially when considering grayscale images where the range of pixel values is between the value 0 to 255. It begins a bit harder to use Binary thresholding with images with more colour channels.

$$D_{i,j} = \begin{cases} 0, & \text{if } D_{i,j} \leq \tau. \\ D_{i,j}, & \text{otherwise.} \end{cases} \tag{3}$$

By summing over these obtained values, the result may make more sense as it is a representation of how many pixels in the image have changed such that their change is significant. Of course this "significance" is determined by the user and the chosen τ.

It is obvious that this is avery simple way of calculating difference between images. To ensure accurate results, the camera has to stay very still otherwise, potentially, every pixel could experience a change in intensity. This can be somewhwat countered by choosing and appropriate τ. However, if looking for finer changes, the chosen τ could remove some of these changes, disabling detection.

In addition, if the images are taken in different lighting environments then that will cause inaccurate indications of difference. Take two potential scenes, exactly the same and untouchedd camera fixed. In the day time, the image captured will have a much higher intensity than an image taken in the evening. This coud cause the whole image to be above the threshold of changed but actually tell us nothing about changes in the scene, if any. The problem with dealing with illumination changes becomes more prominent in surveillance literature. A way to reduce the illumination effect is to perform a normalisation of the pixels based on average intensities and variance. This shall not be explained further but details of it and other detection techniques are highlighted it Radke's paper[10].

Image Ratioing

Image Ratioing is another method to compare images. The general concept is explained (along with other change detection techniques) in Singh's paper *Review Article Digital change detection techniques using remotely-sensed data*[**?**]. Again we have our two images, from before, taken at different times. From here, the pixels' intensities are compared to obtain a ratio:

$$R_{i,j} = \frac{A_{i,j}}{B_{i,j}} \tag{4}$$

The closer this $R_{i,j}$ values is to 1, the more similar the two pixels are in terms of intensity. Thus, if $R_{i,j}$ is quite far from 1, it is a good indicator that there has been change within the image. Again, thresholding can be applied to ths Ratio Image, R, which will leave only those with a significant ratio to be considered. The same steps as image differencing can then be utilised to gain a representation of the overall change in the image.

Existing Projects

LandscapAR

The AR Sandbox

Nowadays, technology is obsessed with being more interactive, especially in the realm of games and education. With the emergence of hardware such as Google Glass and the Oculus Rift, augmented and virtual reality is becoming a hot and exciting field in which to make technological advances. Emphasis has been made on "immersing ourselves" within the game or environment that we are interacting with, in order to get a much more stimulating experience which leaves a great impression upon us. I myself had taken a liking to Augmented Reality (AR) sometime before applying for my M.Eng Computing degree at Imperial College London. This original interest provided most of the motivation for taking up this project.

Investigating Hardware

The rest of my motivation for this project came from the investigation into the current touchless devices and advanced displays on the market and available to developers and consumers alike to play around with. I looked into several of these hardware devices of which I now outline my findings.

LeapMotion

Background & History

The Leap Motion company has been around since 2010, the release of their touchless control/ input device, the Leap Motion, was in 2013 meaning it is a relatively new entrant into the market. However, it has generated a lot interest in this area of software development. The Leap Motion is one of the better performing touchless controls on the market and has a growing community of developers. The Leap Motion has an online app store[1] that features various games, apps as well as tools and utilities produced by developers and available for other Leap Motion owners to purchase and play with.

[1] https://apps.leapmotion.com/

How it works

The Leap motion comprises of 2 infrared cameras and 3 infrared LEDs, these can be seen in figure 6. After being placed flat on a surface, the Leap Motion emits infrared light and captures it back once it is reflected off surfaces, namely your hands. The Leap Motion tracks near field infrared, making the captured images grayscale and able to be shown in the development kit. The device boasts 10 finger, simultaneous tracking within a range of around 8 cubic feet. The capture range is in a hemispherical shape as can be observed in figure 7. This proves to be a more than suitable capture range if you are using the device within the proximity of your computer or laptop. The Leap Motion blog has a very good guide explaining how it operates [2].

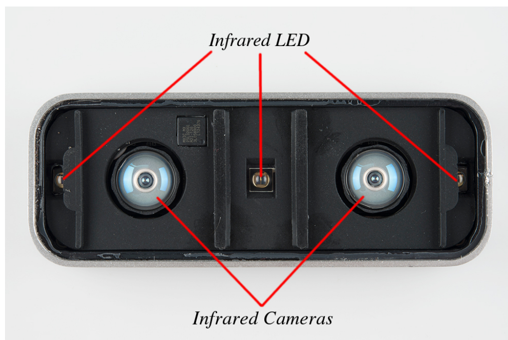


Figure 6: A look inside the Leap Motion sensor

[2]http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

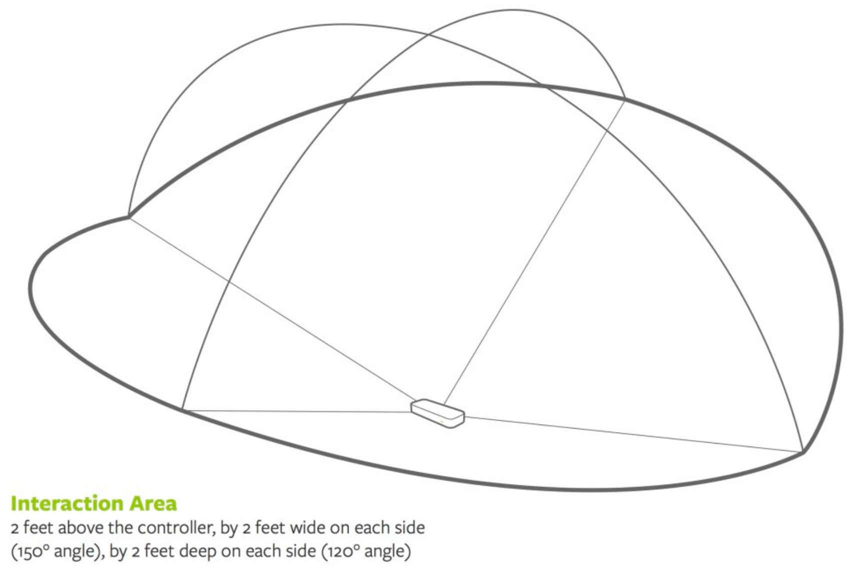


Figure 7: The reported range of detection of Leap Motion

We can interact with the Leap Motion by placing our hands within its capture range. If we were to move our hand past the plane on which the infrared cameras are placed, the device registers that as a selection by the user, equivalent to a mouse click. Any activity behind the plane, i.e. closer to the user, will allow the user to navigate the controls.

My Investigations

I had the chance to play around with the Leap Motion as part of my investigations and background research. There is a moderate community of developers for the Leap Motion at the moment, applications are obtained by purchasing them on their the Leap Motion online store.

There are currently 219 applications for Leap Motion as of January 2015[3], available on the store. Of these, 136 are classed under *Games*, 8 as *Virtual Reality* and 42 classed as *Music/Entertainment* (these categories are not mutually exclusive). We can take this as a representation of what exactly the Leap Motion is currently capable of doing well. Over half of the applications were aames, even more could technically be considered a game.

The other categories of application included Education,Computer Controls, Utilities and Creative Tools. These do not come anywhere near as close as the number of Game apps,

[3]https://apps.leapmotion.com/categories/all

Utilities was the closest with 30 apps under its category on the store. Existing technologies and conventional methods of input such as webcams, mice, touch screen and keyboard are probably still the much faster alternative than navigating with a touchless display for these areas of applications. Using the Leap Motion for these particular uses would probably be much slower and less efficient (and most of the time more infuriating as I have experienced first hand!) than conventional input methods.

There have been a numerous amount of reviews for the Leap Motion, *Tested*, a popular tech youtube channel, had a review concerning the device [4] where they stated the Leap Motion was "Not a mainstream device", "...just a bag of crap; set it on fire". However, *Tekzilla*, another popular Youtube channel which promotes new technologies offered a much better review, albeit, bringing in someone from the company to demo the device [5].

Leap Motion has also paired up with the company Oculus [6] to release their 2nd development kit. Oculus is a company specialising in the production of head mounted devices which are used for virtual reality applications. The two have worked closely to allow the Leap Motion to be mounted onto the Oculus Rift VR headpiece [7]. This has opened up the possibility for a new, wide range of Virtual Reality Games with touchless interaction. There are already examples of such games on the Leap Motion store. I shall not investigate further into the Oculus as my project does not deal with virtual reality.

<u>Limitations and pitfalls</u>

When using the Leap Motion, there were several things I noticed. Although the visuals and applications are very pretty and nice, it does not matter when the Leap Motion decides not to work. As this company and its device are the best, small, double-hand motion and image capture currently out there, there is definitely scope to manipulate what this piece of hardware offers.

Due to the sensors using infrared light and cameras collecting the reflected, emitted light, there are some very clear limitations of the Leap Motion. Infrared light travels in straight lines, as a result, if you occlude your fingers behind one finger, say by arranging your hand such that your thumb faces the ceiling and place this directly above the Leap Motion, it'll be unable to tell what your fingers are doing, bar the one that it can actually see. We can see evidence of this in figure 8 where my hand circled in red is clenched but the Leap Motion momentarily still believes it has a finger sticking out.

[4] https://www.youtube.com/watch?v=ZK5FRPwIWVE

[5] https://www.youtube.com/watch?v=gOtAhU3DIv0

[6] https://www.oculus.com/

[7] https://developer.leapmotion.com/vr

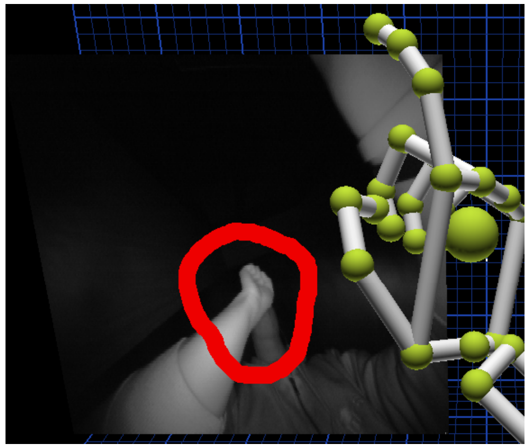


Figure 8: What cameras see vs skeletal hand mapping

One way to try tackle this apparent problem is to use historical data, i.e. the movement of points of the fingers over the past 3 seconds to help estimate the position of any occluded fingers, though if this was implemented is not at an acceptable level yet, especially if time critical, responsive applications require your hands to be in the positions of self occlusion.

In fact,on the Leap Motion website they state that the device software will "...interpret the 3D data and infer the positions of occluded objects. Filtering techniques are applied to ensure smooth temporal coherence of the data.". So infact already do some kind of inference on the likely positions of the fingers and joints in according to what they have managed to captured. Though it does take a few seconds, the Leap Motion can quite accurately, under what I have managed to explore hands on with the device, correct the input to what it actually is in real life.

Another noticeable limitation of the Leap Motion is its precision and ability to deal with movements which require more finesse. Firstly, you cannot even think about clenching or clasping your hands together as the sensor will no longer detect any hands. In addition, it was even apparent through the demo games that the Leap Motion takes a lot of getting used to as trying to perform small actions such as plucking a single petal, as seen in figure 9, was very difficult to get right, at least in my case and many of my friends' cases. It seems minute and delicate gesture inputs aren't captured and expressed very well by the Leap Motion and its current apps.

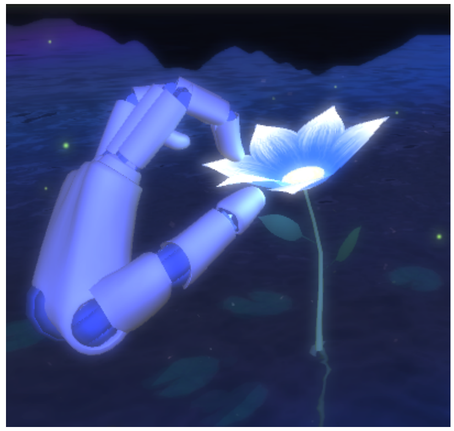


Figure 9: The Demo game for Leap Motion

To use the device effectively, you have to place your hands in a constant state of being parallel to the Leap Motion such that each of your 10 fingers can be captured by the infrared cameras. In addition, when you bring your fingers too close together, the sensor becomes less and less able to distinguish it as 2 fingers or 1, fat, finger, providing further loss of information and inaccurate representation of input, requiring you to be conscious of the placement of your hands and fingers in relation the other hand or fingers.

Another problem arising from the infrared LED cameras is that they can absorb any infrared light, thus in an environment where there may be external sources of infrared such as in the outdoors, the Leap Motion becomes less responsive and finding it hard to properly track the fingers as the light collecting in the camera may first be more than emitted and also will affect different parts of the sensor creating a grayscale image with possibly large white areas. In addition to this problem, when indoors and your Leap Motion is pointing upwards onto a particularly reflective ceiling or one that has an infrared light source, the same inaccuracy happens. A very neat trick, however, is that the Leap Motion is able to detect some discrepancies in the capture of the hands that it can suggest to us when we might want to clean the device from finger smudges.

I gained further insight into the performance of the Leap Motion through the paper *Analysis of the Accuracy and Robustness of the Leap Motion Controller*[7] which looks into the accuracy of the preliminary Leap Motion (release version should perform at least this well) and assesses its accuracy. The guys at Leap Motion claimed finger and movement racking to be accurate within 0.01mm. Subjecting the Leap Motion to average conditions, the paper concluded that although this claim was not likely achievable, a decent 0.7mm accuracy could still be achieved.

In addition there have been a few interesting use case proposals for the Leap Motion. One such piece is explored in the paper *Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller* [4]. It puts forward the idea of creating a human interface with the elderly such that they can carry out "Activities of Daily Living (ADLs)". With the Leap Motion as input to control robotic arms, humans actions can be mimicked by the users such that if they would otherwise not have the strength or coordination to perform an action, they will be able to control the robotic arm very easily by just performing the action over the sensor. Those suffering from Alzheimer's or significant limb/muscle weakness will benefit from this and the paper also talks about correcting hand tremors and shakes from the input to provide a smooth output. Being only $80 per piece of Leap Motion device, it is also cheap alternative, in today's standards, to specialised sensors as well as its size advantage, being smaller than gyroscopes and accelerometers.

Microsoft Kinect V2

Background & History

The Microsoft Kinect(V1) was original released as an extra piece of hardware for Microsoft's Xbox360 in 2010. The Kinect for windows (V2) was released in 2012 and offered a range of improved statistics from its ancestor, V1. Since its introduction into the market, the Kinect has received a lot of attention from developers and there have been many applications for it put forward[8]. Many large companies have caught onto its capabilities and work alongside Microsoft and the Kinect.

In contrast to the Leap Motion, the Kinect V2 offers users the ability to track their whole body with reasonable accuracy. In addition, the V2 can track up to 6 individuals at one time with 25 body joints able to be placed onto the captured images. This opens up a whole new range of applications that the Kinect can be used for and this is apparent in the amount of interest in the Kinect, along with the amount of published papers and success stories of different uses of it. Some of the possible fields this device can be used include Augmented Reality, Entertainment and Games, Retail, Education and much more. The Microsoft Kinect website highlights some of these interesting projects in more detail [9].

The paper "Evaluation of the Spatial Resolution Accuracy of the Face tracking system for Kinect For Windows V1 and V2" [2] explores in more detail the increase in performance and capability of the V2 and shows that is beats V1 in every aspect.

[8] http://www.microsoft.com/en-us/kinectforwindows/meetkinect/gallery.aspx?searchv=entertainment

[9] http://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx

How it works

The V2 allows us to take infrared images along with 1080p colour images. It also has depth sensors to produce depth images. Inside the V2 are 3 infrared emitters along with a colour camera, an infrared camera and some microphones. These internal components can be seen in figure 10. The microphones are along the bottom but will not be used within my project.

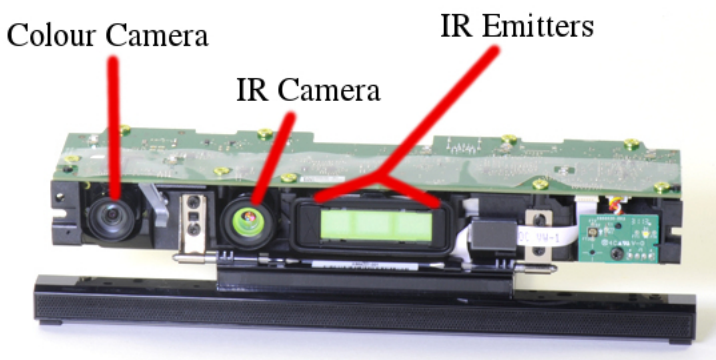


Figure 10: Inside the Kinect V2 for Windows

The Kinect V2 differs from the V1 majorly in the way it produces depth maps. V1 uses structured light while the V2 utilises Time-Of-Flight. Time-of-Flight involves emitting many short bursts of infrared light (strobing it) and then collecting it back through its camera. A very thorough discussion of how this determines the depth of objects in the captured image is present on Daniel Lau's blog [10]. The technique involves splitting a pixel in half and collecting infrared light on the pixel. However, they are on at different times, while one half is on, the other is off and depending on the ratio of how much light is collected by each half (ratio since it accounts for light absorption by objects in the scene), the relative depth of objects can be inferred.

The V2 also accounts for over exposure and saturation of pixels. This scenario occurs when there are alternative sources of infrared light, i.e. in an outdoor environment! The V2 can reset pixel values in the middle of an exposure, also explained in Daniel Lau's blog mentioned above. This then allows it to be used outside where the V1 wouldn't account for this and cause all kinds of strange behaviour; this of course means that the scope for using the V2 and its applications is significantly larger than it ancestor.

My investigations

[10] http://www.gamasutra.com/blogs/DanielLau/20131127/205820/The_Science_Behind_Kinects_or_Kinect_10_versus_20.php

I investigated the Kinect for Windows first hand. In figure 11 you can see an example of the depth sensing capability of V2. On the right, circled in blue, is myself. I sat around 40cm away from the camera. As I highlight in the limitations section below, this is reported as the minimum distance performance can be guaranteed by the V2. My friend, Juto Yu, is circled in yellow, he say slightly outside the view of the camera, giving me a rough indication on how wide the capture region is. The green circled object is a pillar in the central library of Imperial College, it is definitely more than 3m away from the V2 which is reported as the range before objects become "too far". Whether it being detected is because the device can actually handle larger caption depths or due to the pillar being white and thus more reflective will need to be further looked in to.

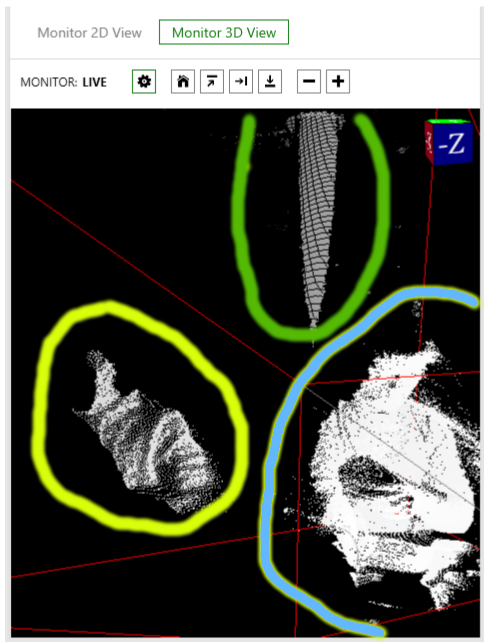


Figure 11: Depth image from Kinect V2

I also looked into the colour camera's performance. The camera has a 1080p HD video taking capability and is of a good quality. In addition, as seen in figure 12. The body tracking is pretty good and can make rather accurate implications of people's limbs even when they are sitting down and are obscured as shown. I was sitting at a desk but the V2 could make some guesses as to where my legs would be. It is also clear that multiple tracking can be done in real time at a speed that is responsive.

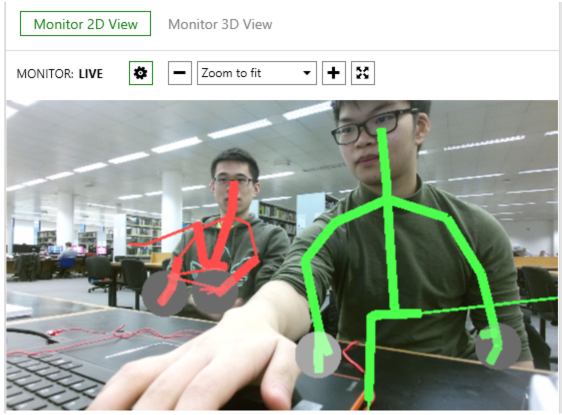


Figure 12: Colour Camera (with body tracking)

The V2 is definitely a great alternative to some of the more expensive depth sensors on the market. As highlighted in the paper "Low-cost commodity depth sensor comparison and accuracy analysis" by Timo Breuer, Christoph Bodensteiner, Michael Arens. Some of the other sensors that out perform it are upwards of 3000 while the V2 costs just around 150. This means that it will be accessible to a much larger audience, something definitely needed if an Augmented Reality game is to be successful.

<u>Limitations and pitfalls</u>

The cameras of the V2 have a set range of values that it works well within. This is reported by Microsoft to be from 0.4m to 3m [11]. Anywhere beyond that seems to be too far to determine the depth of objects. There is also a notion of being too near to the device to! If we are working with the device, say, next to our laptop or in a space nearby, there may be complications with how the sensor performs, anything closer than 0.4m produces unknown behaviour with the depth sensor.

A pitfall I expect to encounter is when developing with the Kinect SDK. I have experience in C++ which is one of the languages that can be used to write Kinect applications. However, it seems a lot of the colour camera tutorials are written in C# which will could cause problems for me if I had to pick up the language or translate it into the C++ equivalents. However, this should only be a small pitfall and cause minor hindrance to the progression of the project.

[11] https://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges

Anticipated End State

With this hard deadline in mind, I need to set a state of the project which I think I would have reached by the deadline. Past this I do not aim to make any significant changes bar running tests for evaluations, which means that my code has to be pretty much bug proof by this date.

Of course I would have hoped to finished all of the objectives I set out in the beginning of this report. However, if things do not progress as quickly as you would like them on any project. I will put more focus into the Augmented Reality part of the project. Integrating the Leap Motion and the touchless manipulation of the created landscape is not necessary for the project to offer a novel gaming experience. The game can lie within the creation of the landscape from a user drawn 2D landscape and have that come to life by using the Kinect.

I would definitely like to reach the point in the project where a someone can create a 3D AR landscape. I expect the bulk of the work will lie within this part of the project as there is a lot to do with image processing to complete this step. The creation of the landscape may also take some time as I am still unfamiliar with what I should use to create this for the user. The next major milestone will be extending this point to provide the created landscape with an illumination model that is equivalent to the scene it is being portrayed within. I expect this also to take up a fair time, however, since I am learning about it within a module of my course right now, hopefully I will be able to grasp its concepts faster than without. Making amendments to the illumination should take a bit of time. Past this point I shall consider as extensions to the project that would be great to implement. Though the investigation and project will concentrate on the two points above.

Extensions

I have two planned extensions for this project immediately on completion. The first of which is that after the render of the first Augmented Reality scene, the user can choose to fix that in place (though rotation and translation of the scene can still take place) and then introduce modifications to the scene in real life. For example, placing in a pencil case, a water bottle or even a scrunched up piece of paper. This will then reflect upon the AR scene. Imagine it as you being a Giant, playing around with the AR world that you have created. The scene should change with the introduction of this new object, this will include occlusions, different shading and light to name the main challenges.
The next extension would be to allow you to interact with the landscape using the touchless input device, Leap Motion. What exactly this will entail has not yet been set in stone but

some ideas I have are the following:

- Allowing the user to add accessories or other landforms onto the scene by selection and direct placement. Imagine it as the Sims or any other simulation game where you can choose to build a house, etc. on the scene.
- Introducing randomly generated content such as miniature people, avatars, objects etc. to traverse the landscape which can then be manipulated and moved around.
- Allowing the user to use view their hands in the scene, kind of like they were playing God, allowing them to interact with the scene, for example if they spread their hands, it could cause rain on the scene.

One other extension which is pretty ambitious would be to track the original feature drawing you provide to generate the first AR image. After that, if the user deforms this image, the scene will deform alongside. This means that extending past drawing on more features, if the user scrunches up the paper, tears it in half or transforms it in some way, allow the scene to change such that it reflects the difference in between the before and after of the feature drawing. For example, tearing the drawing will cause a rip to form in the generated landscape. Deforming the drawing will deform the landscape. In essence, this will be taking feature points of a feature drawing!

Evaluation Plan

The below shall outline my evaluation plan of this project. It is important my evaluation is rigorous and solid so that it proves that what I would've produced at the end is something others can use and build upon. This is because if I am to provide any kind of advances in this field, then what I produce must be proven to be sound. I will be basing my Evaluation plan over the points I outlined in my anticipated end state in my project plan.

Indicators of Success/Benchmarks

Functionality

At the end of this project, I hope that I am able to quickly draw up a feature drawing on a piece of paper within 3 minutes (5 minutes if they want to be interesting and have a nice landscape!). I should then be able to place this in front of the Kinect device and have my created piece of software read the image and produce Augmented Reality landscape within some acceptable time, for example within 1 minute the Landscape should have been created and displayed. This is important because for AR to be effective and even useful in any real life application, it has to be fast and responsive, being close to real time as possible. In addition, since the main motivator of this project is that current AR applications do not look realistic enough. I hope that by the end of this project I can produce a landscape that looks realistic, at least in terms of illumination.

User Testing

For any game to be considered anywhere near success, user testing needs to take place. I intend to test the software on a range of people, from those not technically inclined to the younger generation, including children! Hopefully I can then ascertain the age groups to which this can be marketed to. If the feature drawings require a relatively high amount of detail, then young children are likely to find troubles with getting what they want out of the software. Hopefully through user testing I can also discover faults, possible extensions, improvements and edge cases. I'd like to set the number of test subjects as at least 20 individuals, of which multiple testing will occur.

Responsiveness

To further enforce what I mentioned in Functionality, there will be a whole other set of evaluation criteria I set out to assess the success of my project. The first of which is responsiveness of software, which is what I mentioned above. The software should generate the illuminated 3D landscape within a minute of being presented the feature drawing. An

even better turnover would be ideal, hopefully within the range of 15 seconds as even a minute is long to wait for an Augmented Reality game.
The software should also be able to detect rotation of the feature drawing and also rotate the landscape as appropriate. This would ideally be a lot more responsive than the target above, within the 2 second range. Any more and the movement would look choppy and will not represent an accurate rotation. However, as the illumination model is a lot more advanced, this may or may not be possible, considering some very basic Augmented Reality applications don't even react this fast in the presence of rotation. This rotation should also reflect in the shadows of the landscape.

Testing

Functionality

To give qualitative and quantitative results to the evaluation criteria I have proposed above, first it is very easy to just measure the time it takes for the software to generate the 3D landscape and then see how long it takes from the moment the user presents their feature drawing to the Kinect. How close the performance is to the time boundaries I have outlined above will determine how well the software and project can be deemed a success. However, the rate of success will have an exponential relation to deviance from the set boundary. This is because if it gets too slow, it would cause much anger and frustration to the users of the software because as I have mentioned already within this report, AR and its growth and success heavily depends on being next to real time. To give an idea, if I rotated the scene, then if it took above 5 seconds to reflect the change in the landscape, this is way too slow to be of any use or of any fun. This will be a failure.

User Testing

Being a game and very much a visual project, feedback will be needed from a numerous amount of people due to the subjectiveness of these criteria. Written feedback in the style of questionnaires would be a good way to quantify the performance of my project by the people I will be asking to take part in my tests.

Responsiveness

The responsiveness can be tested against some hard time boundaries, once again with their exponential penalty the more time it is from the the boundary. This can easily be tested by tracking how much time it takes between tested actions. When it comes to the assessing the responsiveness of the illumination change, this can be measured by eye, if the illumination stays consistent with the lighting in the scene, then generally this can be considered a success. If the scene looks realistic and the shadow placement makes sense and obscured areas become visible on rotation, then these are also indicators that it passes the test. The realism of a scene is a lot more subjective and therefore, as mentioned above, the test will range over a set amount of people. If the majority thinks the scene still looks real after a change, we can consider this a success.

Bibliography

[1] 09ggr820. Canny edge detection. Indian Institute of Technology Delhi, Department of Computer Science and Engineeering, March 2009.

[2] Clemens Amon and Ferdinand Fuhrmann. Evaluation of the spatial resolution accuracy of the face tracking system for kinect for windows v1 and v2. volume 6th Congress. Alps-Adria Acoustics Assosiation, 2014.

[3] Ronald T. Azuma. A Survey Of Augmented Reality. Technical report, Hughes Research Laboratories, August 1997.

[4] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, and T. Bock. Intuitive and adaptive robotic arm manipulation using the leap motion controller. In *ISR/Robotik 2014*, volume Proceedings of 41st International Symposium on Robotics, pages 1 – 7. VDE, June 2014.

[5] John Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Pami-8(6):679–698, November 1986.

[6] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern Recognition*, 34:721–725, 2001.

[7] Bartholomus Rudak Frank Weichert, Daniel Bachmann and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. Technical report, Department of Computer Science VII, Technical University Dortmund, May 2013.

[8] Wendy E. Mackay. Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers. Technical report, Department of Computer Science, Universit de Paris-Sud.

[9] Raman Maini and Dr. Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International Journal of Image Processing (IJIP)*, 3(1):1–12.

[10] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. Technical report, Rensselaer Polytechnic Institute.

[11] Rashmi, Muhesh Kumar, and Rohini Saxena. Algorithm and technique on various edge detection: A survey. *Signal and Image Processing: An International Journal*, 4(3):65–75, June 2013.

[12] S. Reed, O. Kreylos, S. Hsi, L. Kellogg, G. Schladow, M.B. Yikilmaz, H. Segale, J. Silverman, S. Yalowitz, and E. Sato. Shaping watersheds exhibit: An interactive, augmented reality sandbox for advancing earth science education. University of California, Davis and American Geophysical Union (AGU) Fall Meeting 2014, 2014.

[13] Paul L. Rosin. Thresholding for change detection. Technical report, Brunel University.

[14] G. T Shrivakshan and Dr. C Chandrasekar. A comparison of various edge detection techniques used in image processing. *IJCSI International Journal of Computer Science Issues*, 9(5):269–276, September 2012.