

# **AR Campus Tour**

**Patrick Geoghegan**

## **A Dissertation**

Presented to the University of Dublin, Trinity College  
in partial fulfilment of the requirements for the degree of

**Master of Science in Computer Science (Augmented and  
Virtual Reality)**

Supervisor: Aljosa Smolic

August 2018

# Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

---

Patrick Geoghegan

August 30, 2018

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

---

Patrick Geoghegan

August 30, 2018

# Acknowledgments

I would like to thank my mom, my dad, my sister and Adrian for all their love and support over the past year, and for proof-reading this paper.

I would also like to thank Professor Aljosa Smolic for teaching me the wonders of Computer Vision and Augmented Reality, and for his guidance and supervision over the course of this project. I also thank Sebastian Lutz and Tejo Chalasani for their guidance as well.

Finally, I would like to thank my friends and colleagues from within the college community, as well as outside, for a year of valuable lessons, learning and laughter.

PATRICK GEOGHEGAN

*University of Dublin, Trinity College  
August 2018*

# AR Campus Tour

Patrick Geoghegan, Master of Science in Computer Science  
University of Dublin, Trinity College, 2018

Supervisor: Aljosa Smolic

With the growing interest and use of Augmented Reality in business and academia, many studies have been conducted into the technology's use in the tourism industry. Using publicly available tools and frameworks, this paper maps out the design and implementation of a tourism AR application, called the AR Campus Tour. This application is built to guide users around the grounds of Trinity College Dublin, and direct them to various famous landmarks. This paper also examines different algorithms that can be used to create a quick and accurate system.

# Summary

The paper is laid out as follows: first the background research will be discussed. Topics will include benefits of technology in the tourism industry, the existing research on AR in the tourism industry, as well as a literature overview of image recognition algorithms. Next, the design of the application will be laid out with each component discussed in detail. Following this, the actual implementation of the application will be explained, including a list of the technologies used and a walk through of the application. The results of the algorithm tests will then be presented and discussed. Finally, the conclusion will be given followed by the limitations and further work that can be completed on the AR Campus Tour application.

# Contents

|   |             |
|---|-------------|
| <b>Acknowledgments</b>                                  | <b>iii</b>  |
| <b>Abstract</b>   | <b>iv</b>   |
| <b>Summary</b>  | <b>v</b>    |
| <b>List of Tables</b>                                   | <b>viii</b> |
| <b>List of Figures</b>                                  | <b>ix</b>   |
| <b>Chapter 1 Introduction</b>                           | <b>1</b>    |
| <b>Chapter 2 Background</b>                             | <b>3</b>    |
| 2.1 Technology in Tourism . . . . .                     | 3           |
| 2.2 Augmented Reality in Tourism . . . . .              | 6           |
| 2.2.1 Definition of AR . . . . .                        | 6           |
| 2.2.2 Uses of AR . . . . .                              | 6           |
| 2.2.3 Combination with Mobile . . . . .                 | 7           |
| 2.2.4 Effects of AR . . . . .                           | 8           |
| 2.2.5 Mobile Hardware Restrictions . . . . .            | 9           |
| 2.2.6 Marker Vs. Markerless . . . . .                   | 9           |
| 2.3 Image Recognition . . . . .                         | 10          |
| 2.3.1 SIFT . . . . .                                    | 10          |
| 2.3.2 SURF . . . . .                                    | 16          |
| 2.3.3 ORB (Orientated FAST and Rotated BRIEF) . . . . . | 20          |
| 2.4 Feature Matching . . . . .                          | 22          |

|                   |                                    |           |
|-------------------|------------------------------------|-----------|
| 2.4.1             | Brute-Force Matching . . . . .     | 22        |
| 2.4.2             | FLANN-Based Matching . . . . .     | 23        |
| <b>Chapter 3</b>  | <b>Design</b>                      | <b>24</b> |
| 3.1               | Pipeline . . . . .                 | 24        |
| 3.2               | Mobile Device System . . . . .     | 25        |
| 3.3               | Database System . . . . .          | 25        |
| 3.4               | Image Recognition System . . . . . | 27        |
| <b>Chapter 4</b>  | <b>Implementation</b>              | <b>28</b> |
| 4.1               | Technologies Used . . . . .        | 28        |
| 4.2               | Technical Architecture . . . . .   | 29        |
| 4.3               | UI/UX Design . . . . .             | 30        |
| 4.4               | Application Walkthrough . . . . .  | 31        |
| <b>Chapter 5</b>  | <b>Results</b>                     | <b>35</b> |
| <b>Chapter 6</b>  | <b>Conclusions</b>                 | <b>43</b> |
| <b>Chapter 7</b>  | <b>Further Research</b>            | <b>44</b> |
| <b>Appendices</b> |                                    | <b>50</b> |



# List of Tables

|     |  |    |
|-----|--|----|
| 2.1 | Example Scale Space (smaller images are scaled up for demonstration purposes) . . . . .                                | 13 |
| 5.1 | Algorithm testing results - BF means using the Brute Force feature matcher, FL means using the FLANN matcher . . . . . | 36 |
| 5.2 | Algorithm testing results - SIFT is limited to 500 features and SURF has a hessian threshold of 2000 . . . . .         | 40 |

# List of Figures

|      |  |    |
|------|--|----|
| 2.1  | Smart Tourism and its layers and components . . . . .  | 4  |
| 2.2  | Experience Typology Matrix: linking technology and co-creation . . . .                                       | 5  |
| 2.3  | Experience hierarchy . . . . .   | 5  |
| 2.4  | Template Matching . . . . .  | 11 |
| 2.5  | Approximating Laplacian of Gaussian using the Difference of Gaussian<br>method on the first octave . . . . . | 12 |
| 2.6  | Keypoint Detection . . . . .   | 14 |
| 2.7  | Calculating a features orientation using a histogram . . . . .   | 16 |
| 2.8  | Generating feature descriptors [32] . . . . .  | 17 |
| 2.9  | Box Filters for Approximating the Laplacian of Gaussian in SURF [33]   | 18 |
| 2.10 | Scaling the box filters [33] . . . . .   | 19 |
| 2.11 | Orientation Assignment in SURF [33] . . . . .  | 19 |
| 2.12 | FAST Corner Detection [36] . . . . .   | 21 |
| 3.1  | Pipeline . . . . .   | 25 |
| 3.2  | Mobile Device System . . . . .   | 26 |
| 3.3  | Database System . . . . .  | 26 |
| 3.4  | Recognition System . . . . .   | 27 |
| 4.1  | Technical Architecture Diagram . . . . .   | 30 |
| 4.2  | UI Elements of AR Campus Tour . . . . .  | 31 |
| 5.1  | Feature detection time against complete image recognition process time                                       | 38 |
| 5.2  | Number of Features Detected against Time Taken to perform image<br>recognition . . . . .                     | 39 |

|     |   |    |
|-----|---|----|
| 5.3 | Number of Features Detected against Time Taken to perform image recognition separated by query image . . . . .        | 39 |
| 5.4 | Number of Features Detected against Time Taken to perform image recognition for one query image . . . . .             | 40 |
| 5.5 | Features detection time against feature matching time in SIFT (BF, 500)   | 41 |
| 5.6 | Features detection time against feature matching time in SURF (BF, 2000) . . . . .                                    | 41 |
| 5.7 | Normal image on the left, compressed image on the right . . . . .   | 42 |
| 1   | On start - the loading message in the Tour Select Menu indicates an ongoing API request . . . . .                     | 52 |
| 2   | API request was successful - the Tour Select Menu asks users to select a tour . . . . .                               | 53 |
| 3   | Selecting a tour . . . . .  | 54 |
| 4   | A tour has been selected - the Map Button has become active and the Next Landmark Icon has been initialized . . . . . | 55 |
| 5   | Search is unsuccessful - the Nothing Marker has appeared . . . . .  | 56 |
| 6   | Search is successful - the AR Landmark Marker has appeared and the Next Landmark Icon has been updated . . . . .      | 57 |
| 7   | Map is active . . . . .   | 58 |
| 8   | Map after a successful search - a green tick appears above the landmark position marker . . . . .                     | 59 |
| 9   | Database Images of Landmark A - Sphere Within Sphere . . . . .  | 60 |
| 10  | Database Images of Landmark B - Walton Memorial Statue . . . . .  | 61 |
| 11  | Database Images of Landmark C - Statue of Lecky . . . . .   | 62 |
| 12  | Database Images of Landmark D - The Campanile . . . . .   | 63 |
| 13  | Database Images of Landmark E - Statue of Salmon . . . . .  | 64 |
| 14  | Query Images . . . . .  | 65 |

# Chapter 1

## Introduction

The world has seen a huge growth of interest in technologies such as Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR). Until recently, VR was the most popular of these computer graphics and vision technologies. However, more and more study is being conducted into AR, both academically and commercially. This interest was arguably sparked when the successful mobile game Pokemon Go brought AR to the masses. AR will soon expand into many industries around the globe as investigation into its uses has already begun in health care, facilities support and, the subject of this project, tourism.

Tourism is one of the world's biggest industries and contributes massively to the global economy each year. The impact of its contribution affects many people and businesses, including accommodation, transport, entertainment, restaurants, shopping and attractions. As will be discussed in this paper, tourists are always on the look out for new experiences and the industry itself is looking for new technologies to enhance attractions. AR has the ability to satisfy both needs.

The goal of this project is to build an AR Campus Tour application of Trinity College Dublin. The application will recognise different points of interest and overlay information on the device regarding the point of interest. Additionally, the application will provide a navigation function that will guide users to the next landmark on the tour, again through overlaid information. There are multiple methods through which the identification functionality can be implemented, and this paper investigates some algorithms for their suitability (i.e speed and accuracy) for a tourist application such as

this one. As a result of this project, a framework for building AR tourism applications will be created which will contribute to, and build on the existing research of the use of AR in tourism.

# Chapter 2

## Background

### 2.1 Technology in Tourism

There are plenty of studies that chart the use of technology in tourism through the years. In the late 90s, the focus of adopting technology in tourism was the strategic advantage that it provided over competitors [1]. An example of this is selling tickets through an online store to reach more customers worldwide. However, around the same time, Pine and Gilmore [2] identified the consumer's need for an experience as opposed to a regular product or service. As a result, studies began shifting away from technology as a competitive advantage to its usefulness in attracting more consumers by creating new experiences. Technological integration has come in many forms, including the Web 2.0, virtual worlds [3], blogs [4] and social media [5].

In the modern age of smart-products and smart-services, tourism has also seen an upgrade to smart-tourism. According to Gretzel et al.[6], smart-tourism is defined as:

Tourism supported by integrated efforts at a destination to collect and aggregate/harness data derived from physical infrastructure, social connections, government/organizational sources and human bodies/minds in combination with the use of advanced technologies to transform that data into on-site experiences and business value-propositions with a clear focus on efficiency, sustainability and experience enrichment.

This definition has multiple layers and components that can be summarised by Figure 2.1. The layer of main interest in this project is the Smart Experience Layer.

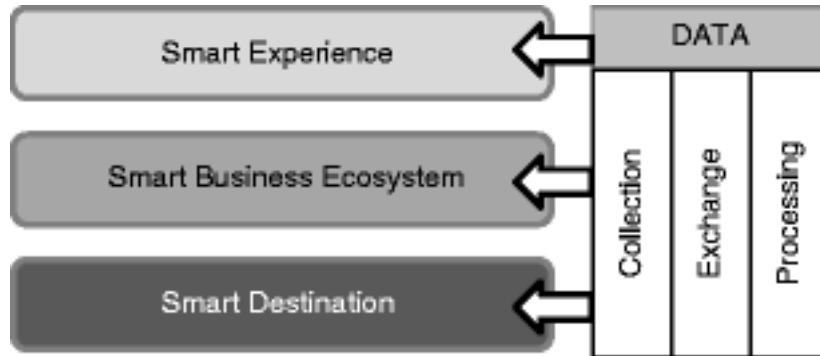


Figure 2.1: Smart Tourism and its layers and components

This layer represents the enhancement of the tourism experience through the use of technology. Enhancements include increased personalisation, context-awareness and access to real-time data [7]. The goal of this project is to provide a smart experience for tourists using a modern technology, augmented reality.

According to Neuhofer et al. [8], technology is one of two key components that create a good tourism experience. The other component is co-creation, which is defined as the joint creation of value by the customer as well as the company [9]. In the case of tourism, this refers to the tourist in playing a primary part in creating the experience. The most notable example of this is the Web 2.0s open, multimedia tools including blogs, videos and images that enables the creation and global sharing of custom content on the tourism experience [10]. The Experience Typology Matrix [8] suggests that the levels of technological integration and co-creation affect what type of experience is being provided (Figure 2.2). This project aims to create a tourism application with high levels of technological integration, with the potential for high co-creation, thus creating a newly enhanced experience, according to the Experience Typology Matrix. Additionally, this project hopes to create a Technology-Empowered Experience as detailed by the Experience Hierarchy shown in Figure 2.3 [8]. This means that technology is not used as a support to the experience but rather, is an integral part of the experience. For example, this project uses augmented reality as part of a tourism application. Without augmented reality the application cannot function. Thus, technology is an integral part to this application.

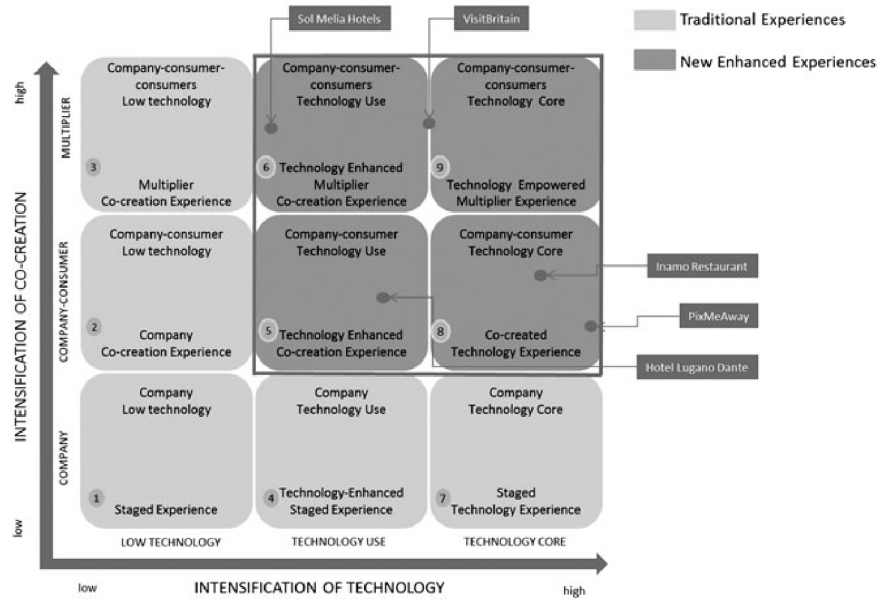


Figure 2.2: Experience Typology Matrix: linking technology and co-creation

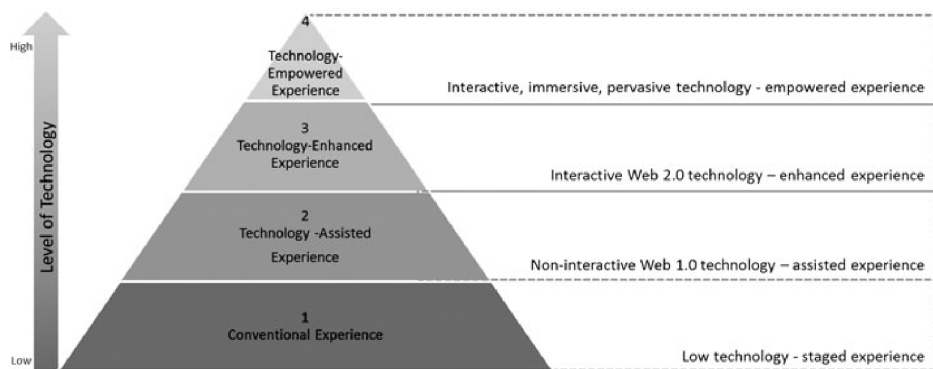


Figure 2.3: Experience hierarchy



## 2.2 Augmented Reality in Tourism

### 2.2.1 Definition of AR

A formal definition for Augmented Reality has not been established. Across different academic papers, definitions of AR focus on a particular aspect of the technology. Zhang [11] defines AR as a “three-dimensional scene where virtual objects are superimposed on [a] real scene”. Chen [12] gives a similar definition calling AR an “environment” wherein both virtual reality (VR) and real-world exists. The author explains that virtual data is superimposed onto the real-world. Other definitions separate AR from VR and define the two as separate technologies. In the context of tourism, Jung et al.[13] describe AR as a natural experience, enhanced with overlaid digital content, whereas VR is a fully digital, immersive experience. For this project, the definition of AR will be a combination of elements from the other definitions above:

Augmented Reality or AR is the creation of an environment, where digital content, including virtual objects and data, is superimposed onto reality in real-time, through the use of a device.

The main goal of AR in the tourism industry is to provide an enhanced experience for tourists, also known as an augmented tourism experience [14]. This augmented tourism experience is defined as a “complex construct which involves the emotions, feelings, knowledge and skills resulting from the perception, processing and interaction with virtual information that is merged with the real physical environment surrounding the tourist”. To explain how AR achieves this, the capabilities and potential uses of AR must be examined.

### 2.2.2 Uses of AR

The possible applications of AR in tourism has already been examined by some researchers. Chen [12] describes an application whereby users can look at a particular point of interest (e.g. a building) and the application will display a historical picture taken of the building. Similarly, Cianciarulo [15] uses AR to vitalise exhibits of historic crafting tools and stations in Italy. For example, a shoemaker’s bench could be identified by the AR application and an interview with old shoemaker would be shown

to the user, describing how this piece of old technology was operated. From these examples, we see that AR has the potential to bring history to life by adding media such as documents, images and videos to places and objects. Another popular use of AR is routing and navigation. It is possible to superimpose some navigation information onto the real world that will guide tourists to the next point of interest. An application proposed by Umlauft et al. [16] enables users to search for a landmark and the application would provide a map as well as textual information guiding the user along the path to the specified location. Although this application did not implement AR, this feature could easily be implemented by adding virtual text boxes or virtual arrows in an AR application. Another way of providing an augmented tourism experience is through implementing interactive AR objects. For example, giving users the ability to tap or click on the object to provide more information about points of interest. Kourouthanassis et al. [17] designed the application CorfuAR, where AR icons are placed around the island of Corfu. By clicking on these icons, additional information about the icon's corresponding point of interest will appear. Adding these interactable AR elements, tourists can interact with points of interest in a new way.

### **2.2.3 Combination with Mobile**

AR does not reach its full potential unless combined with mobile technologies. The concept of Mobile Augmented Reality (or MAR) was presented by Höllerer and Feiner [18]. MAR uses mobile communication and location-based services (e.g. GPS) to take AR out of the desktop and confined spaces. It increases the accessibility of AR by making it available on portable devices such as smart phones and tablets. Typically, a MAR application will use the device's camera to capture the real environment and superimpose some object or data on the cameras output [19]. However, by using mobile communication, GPS and other hardware in the devices, more complex MAR applications can be built. For example, adding the ability to connect to a server or database to access more information or media, or the GPS could be used to build an application that focuses on the user's location (for example, the popular Pokemon Go game).

## 2.2.4 Effects of AR

There are studies that have examined tourists' reactions and responses to the use of AR in tours. Jung et al. [13] investigated the effects of AR on tourists' experiences using the Social Presence Theory [20]. In the context of AR, Social Presence Theory discusses the social interactions between the user and other beings (synthetic or living) in the augmented environment and how this links to the user's immersion in the environment. To summarise, the less the users perceive artificial, the stronger the social presence [21], [22]. Jung et al. [13] show that environments with strong social presence can be built using AR. Following this, the authors found that social presence has a direct positive impact on entertainment experience and entertainment experience has a direct positive impact on visitor experience. Thus, using AR creates a positive, entertaining experience for tourists.

Chen [12] designed an AR application that guided users around the historical landmarks of Oslo. Users of the application found it easy to use and felt that their tourism experience was enhanced. Users also found it convenient to look up historical images and information about points of interest while the camera was pointed towards them. Finally, users were also very enthusiastic about keeping a record of the places they visited. Overall, there was a general positive attitude from users towards the AR application.

Cranmer et al. [23] examined the benefits of AR for the tourism site rather than for the users. The authors highlighted the uses of AR in a Tin Mine Museum and concluded that AR can increase tourist attraction sustainability. Firstly, AR applications are a solution to the challenge of retaining valuable and authentic first-hand knowledge of retired mine workers. At peak times, the application eased the pressure of not having sufficient staff positioned around the museum to interpret exhibits. The introduction of AR increased visitor appeal by "enticing interest and demonstrating site advancement". Finally, when asked, users responded that AR encouraged them to spend more time and money in the site and increased their likelihood of visiting other sites with the technology implemented. He et al. [24] extends this research with their study that shows that visitors willingness to pay more is at its highest when museums combine having a high virtual presence (e.g. using AR to depict an immersive background to a painting) with dynamic verbal information (e.g. playing a voice-over or audio file

explaining the history of the painting), as opposed to dynamic visual information (e.g. adding 2D dynamic visual cues to the painting).

### **2.2.5 Mobile Hardware Restrictions**

Although AR is best used when combined with mobile technologies and devices, there are some limitations when using mobile devices. The major issue is mobile hardware. The relatively low CPU capabilities and RAM capacity, camera quality and WiFi or mobile network connectivity must be considered when building an AR tourism application [25], [26]. To maximise the CPU and RAM capabilities, many applications offload the image processing to an external server. Fatima et al. [27] designed a mobile travel guide application based on image recognition. As part of their implementation, images are sent to a server that handles them using the OpenCV (Open Source Computer Vision Library) and sends the results back to the user's device. Similarly, Gui et al. [28] built an image recognition application and Zhang [11] built a mobile AR game based on image recognition, where both applications offload the image recognition work to an external server. This approach is the best and most popular way of implementing an AR application that requires some image processing. However, the application becomes dependent on the device's WiFi or mobile network connectivity.

### **2.2.6 Marker Vs. Markerless**

There are two forms of AR recognition and display: marker and markerless. Marker-based AR uses special images such as barcodes or QR Codes to identify where a virtual object should be placed. With this method, marker identification is almost instant, and the position and rotation of the virtual object is easily obtained, in terms of tourism, it requires that markers be placed near exhibits and must be scanned at short distances, which could be inconvenient for users [29]. Alternatively, there is the markerless approach, whereby real-world objects, scenes or data are used as the base for AR objects. This achieves the effect of blending the virtual and reality and is more practical for tourism sites [29]. When using the markerless approach, AR content can be triggered through image recognition or through the devices GPS and location data [30], or a combination of both [28]. In the case of the combination, GPS data is used to narrow down the number of images that the image recognition algorithm must

search through by identifying which scenes are close enough to the user. The result is a quicker overall scene identification process and less waiting time for the user.

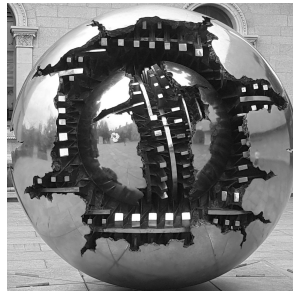
## 2.3 Image Recognition

To identify landmarks in images, the initial approach was to use Template Matching. Template Matching investigates if some input image contains a view of some feature [31]. The view of the feature is captured in the template image. The template image slides over the input image, performing a comparison with a small section of the input image. This process is known as a 2D convolution. The output is a greyscale image, where each pixel represents how similar the area around that pixel is to the template image. An example of template matching is shown in Figure 2.4. The red box shows where the template has been best matched and the accuracy of the match.

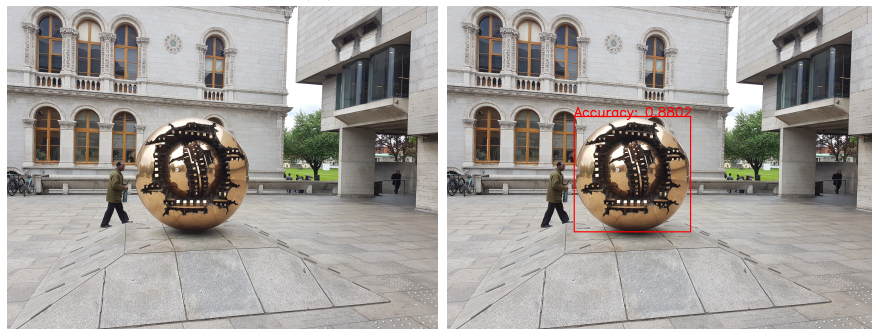
The benefit of this method is that it is quick. This fast return is desirable in an application such as this, because it means that users are not waiting a long time after they have requested information on a landmark. However, there are many issues with the template matching algorithm. Firstly, the algorithm is neither rotation-invariant nor scale-invariant. If either the template or input image is rotated by any number of degrees or is scaled by a factor, the accuracy of the algorithm falls. This is not suited to a tourism application as it is highly likely that users will take pictures of a landmark that is either zoomed or at a slight angle. Secondly, an issue arises in identifying the template image. If the user's image is chosen as the template image, difficulties will arise when using the whole image as the template because if the landmark only takes up a small portion of the template image, it will be difficult to get an accurate result. Additionally, if the user's image captured more than one landmark, the algorithm may return information about a different landmark to the one the user wanted to view.

### 2.3.1 SIFT

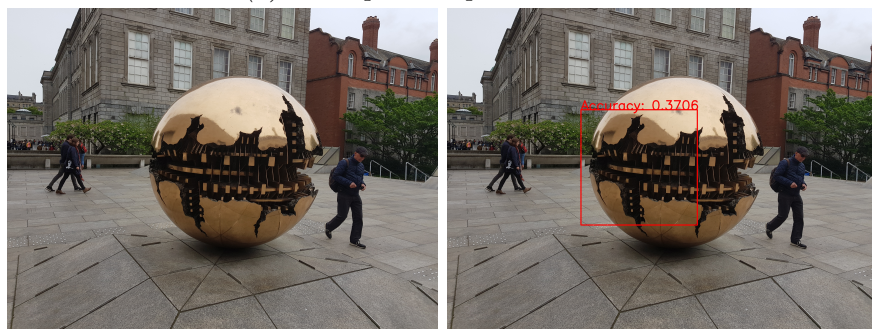
An improved approach to identifying landmarks is using SIFT feature detection. SIFT, or Scale-Invariant Feature Transform, proposed by Lowe [32] is an approach that uses corner detection in images, and handles images of different scale and rotation. The SIFT feature detection algorithm consists of the following steps:



(a) Example Template



(b) Example Template Match 1



(c) Example Template Match 2

Figure 2.4: Template Matching



*Figure 2.5: Approximating Laplacian of Gaussian using the Difference of Gaussian method on the first octave*

**Scale Space** The preparation step of SIFT is to create what is known as a scale space. This process involves doubling or halving the size of the image and blurring it a number of times. This creates a table of images where each column (known as an octave) is a group of images that are the same size but different levels of blur, and each row (known as a scale) is a group of images that are different sizes but equal levels of blur. In the first octave, the images are twice the size of the original. The second octave images retain the size of the original image. For every octave after the second, the size of the images are half the size of those in the previous octave. An example of the scale space is presented in Table 2.1. The number of scales and octaves is left to the developer, however 4 octaves and 5 scales are the recommended amount [32]. The blurring is performed using a Gaussian Blur operation.

**Laplacian of Gaussian Approximation** The Laplacian of Gaussian (LoG) is used to identify the corners and edges in images. First the image is blurred slightly and then the second order derivative (or the “laplacian”) is calculated on the image. The second order derivative is sensitive to noise, however the blurring stabilises the second order derivative by smoothing out noise. Performing the LoG operation is computationally expensive and so an approximation called the Difference of Gaussian (DoG) is used instead. The DoG operation takes the octaves produced in the previous step and calculates the DoG for each pair of consecutive scales using a simple subtraction. The DoG for the first octave in the example scale space is show in Figure 2.5. This process is completed for each octave.

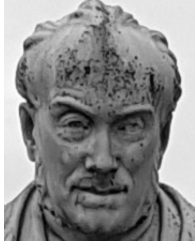
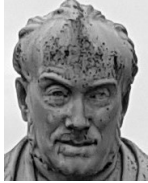


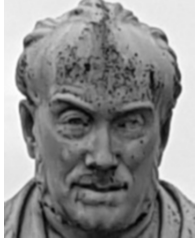



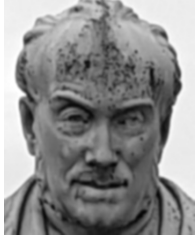
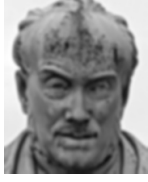


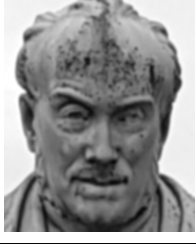
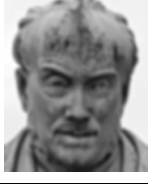


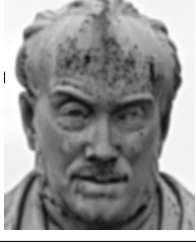
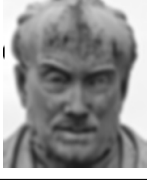


| Scales       | First Octave  | Second Octave   | Third Octave  | Fourth Octave   |
|--------------|---|---|---|---|
| First Scale  |    |    |    |    |
| Second Scale |    |    |    |    |
| Third Scale  |   |  |  |  |
| Fourth Scale |  |  |  |  |
| Fifth Scale  |  |  |  |  |

Table 2.1: Example Scale Space (smaller images are scaled up for demonstration purposes)



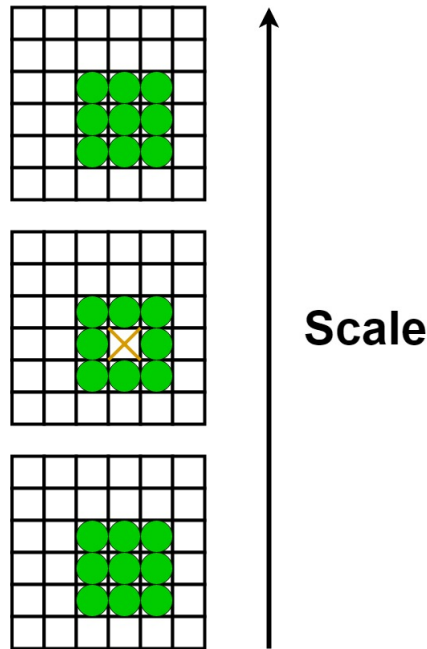


Figure 2.6: Keypoint Detection

**Keypoint Detection** To identify keypoints in the image, the maxima and minima must be located in the images produced after the DoG operation. In each octave of DoG images, we cycle through the pixels of the inner images (i.e. all images except the first and last), and perform a check on the neighbouring pixels as well as the corresponding pixels in the scales above and below the current image. If the current pixel's intensity is greater or lesser than that of all the neighbouring pixels, then it is marked as an approximate maxima or minima. Figure 2.6 illustrates this process. Each large grid represents a scale of an image in one octave. The subsections of the grids represent a pixel. The orange 'X' marks the current pixel and the green circles mark the pixels that the keypoint is compared to. If the keypoint has a high or low enough intensity, it is marked only as an approximate maxima or minima because it is likely that the true maxima and minima lies somewhere in between the pixels. However, using a Taylor Expansion, subpixel data can be created and the true maxima and minima can be calculated, increasing the stability of keypoint detection step.

**Removal of Low Contrast Features** Now that we have produced many keypoints, we need to remove the keypoints that are along an edge or do not have enough contrast.

To remove low contrast keypoints, we calculate the magnitude of the pixel's intensity at that keypoint. If the intensity is lower than a threshold, it is removed. This often involves getting the intensity of the subpixels generated in the previous step. To remove edges, two perpendicular gradients are calculated based on the data surrounding the keypoint's pixel. If both gradients are small, then the area is a flat region and that keypoint is removed. If one gradient is large and the other small, this indicates an edge. Keypoints that are edges are also removed, as keypoints that are corners are the most distinct features. A keypoint is determined to be a corner if the two gradients are large. In this case, the keypoint is kept.

**Keypoint Orientation** To ensure rotation-invariance, the orientation of the remaining keypoints must be calculated. To calculate the orientation, a region around the keypoint's pixel is taken, the size of the region depends on the scale of the image (i.e. how blurred the image is, rather than the size). The higher the blur, the bigger the region. For each pixel in this region the gradient magnitudes and orientations are calculated. Then a histogram is created with 36 bins (of 10 degrees) as shown in Figure 2.7. The magnitude values are placed into one of the bins depending on the orientation. For example, a pixel with the orientation of 43.98 degrees and a magnitude of 3 would be placed into the 40 - 49 bin, and that bar would increase by 3. Once this is complete for all pixels, the keypoint will be assigned the orientation of the highest bar in the histogram. However, if any other bar is above 80% of the highest bar, a new keypoint is created with the same position and scale data as the original but is assigned the orientation of the other bar.

**Feature Descriptor** The final stage of the SIFT algorithm is to create a unique fingerprint that identifies each keypoint. To do this a 16 x 16 grid is created around the keypoint, which is divided into 4 x 4 windows. Again, this stage works with subpixel data and therefore the keypoint lies at the very centre of the grid. Within each of the 16 grid sections, the magnitude and orientation is calculated in a similar way to keypoint orientation method, however, the histograms have only 8 bins of 45 degrees. The amount that is added to the histogram is also reliant on the subpixel's distance to the keypoint. Subpixels that are further away from the keypoint have less weighting to the overall fingerprint. The weight is calculated using a Gaussian weighting function.

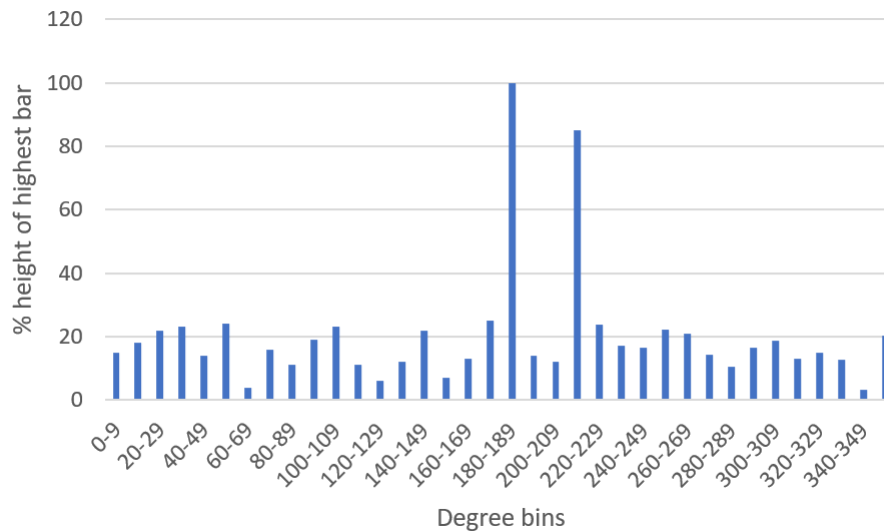


Figure 2.7: Calculating a features orientation using a histogram

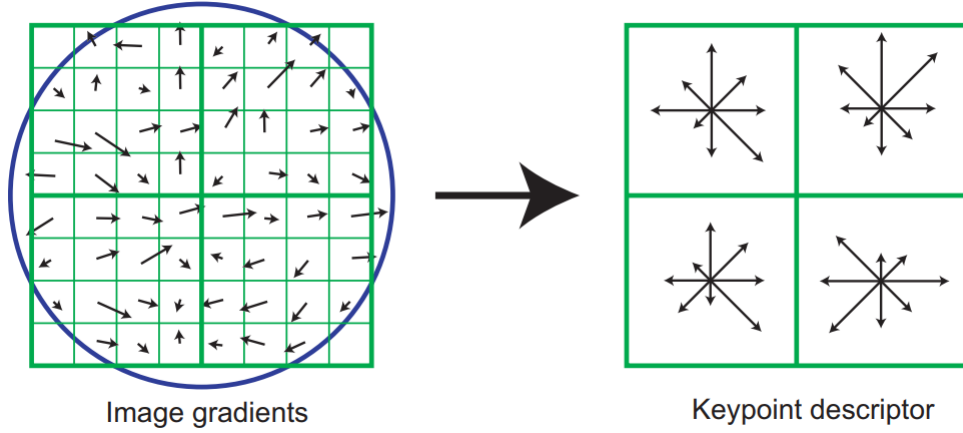
Since there is a 4 x 4 grid with each section containing a 8-bar histogram, the resulting fingerprint is a normalised 128 variable vector. Figure 2.8 shows how feature descriptors are produced using a 8 x 8 grid divided into 2 x 2 windows.

The OpenCV library has SIFT functions for feature detection, so all the steps above are readily available. SIFT only detects features and does not match them. A separate feature matcher must be implemented. Two feature matchers are tested as part of this project. They will be outlined in section 2.4.

### 2.3.2 SURF

The SURF (or Speeded-Up Robust Features) algorithm was published in a paper by Bay et al.[33] and, as the name suggests, it is a quicker version of the SIFT algorithm. SURF makes multiple improvements to the SIFT algorithm which are discussed in this section.

**Integral Image** SURF is able to achieve a speed-up in feature detection by making use of integral images, also known as summed-area table [34]. In summary, an integral image  $I(x, y)$  is an image where each point stores the sum of all pixels in a rectangular



*Figure 2.8: Generating feature descriptors [32]*

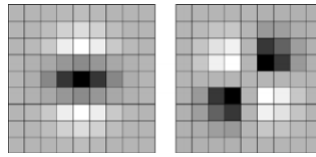
area between the origin and  $(x, y)$ .

$$I(x, y) = \sum_{i=0}^x \sum_{j=0}^y p(i, j)$$

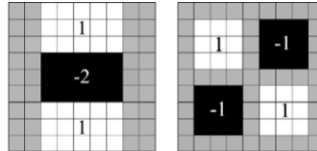
where  $p(i, j)$  is the intensity value of the pixel at  $(i, j)$ . Once the integral image is formed, the sum of intensities in any rectangle can be found with simple addition and subtraction. For example, for some points  $A = (x_0, y_0)$ ,  $B = (x_1, y_1)$ ,  $C = (x_2, y_2)$  and  $D = (x_3, y_3)$ , the sum of intensities is:

$$I_{sum} = I(D) + I(A) - I(B) - I(C)$$

**Blob Detection** SIFT uses Difference of Gaussian to detection potential keypoints, but SURF takes a different approach. Based on the Hessian Matrix, surf uses a blob detector. The goal is to find the determinant of the Hessian matrix, which is an approximation of the Laplacian of Gaussian (for finding edges and corners). The Hessian determinate is approximate by performing a convolution on the image using the box filters pictured in Figure 2.9. The calculation of the convolution is also made easier with the use of the integral image. An interesting benefit to this method of feature detection is that the sign of the trace of the Hessian Matrix will tell if the feature is black with a white background or vice versa. Thus, an improvement in the feature matching phase can be made, in that we only match features of the same colour.



(a) Box Filters



(b) Discretized Box Filters for Approximation and Speed Up

Figure 2.9: Box Filters for Approximating the Laplacian of Gaussian in SURF [33]

**Scale Space** Similar to SIFT, SURF also creates a scale space to ensure scale-invariance. However, unlike SIFT, SURF increases the size of the images with every octave. The same convolution can be performed on the larger images by increasing the size of the box filters. When increasing the size of box filters, it must be ensured that while the size of the kernels is being increased, the lobes must also be correctly scaled. An example of this is shown in Figure 2.10. From this point on until the orientation assignment, the SURF algorithm follows similar steps as SIFT: the local maxima are found, and the true location of the maxima are found using subpixel data.

**Orientation Assignment** SURF uses a method called Haar Wavelet [35] to assign orientation. The Haar Wavelet is applied to the image in a horizontal and vertical direction. The integral image is also used here to simplify the calculations. Then a region around the keypoint is taken. The size of this region depends on the scale of the image. Gaussian weighting is applied to the Haar Wavelet responses so that the responses that are closer to the keypoint have more of an effect on the orientation assignment. The responses are then plotted in a space as shown in 2.11. The orientation is calculated through the sum of all the responses.

**Feature Descriptors** When describing features in SURF, a region is taken around the keypoint and divided into 4 x 4 windows (similar to SIFT). The size of the region again depends on the scale of the image. For each window, the responses to the Haar

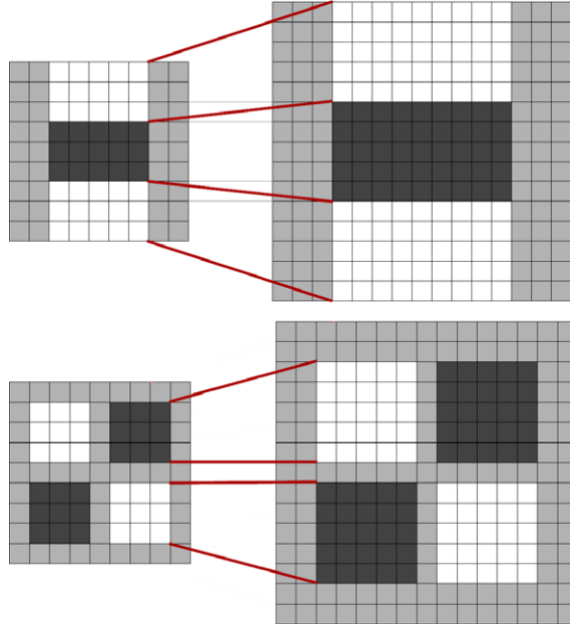


Figure 2.10: Scaling the box filters [33]

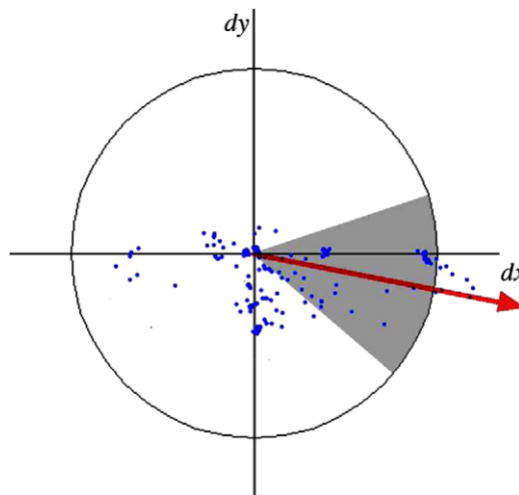


Figure 2.11: Orientation Assignment in SURF [33]

Wavelets that were performed in the previous section are used to create a 4-variable vector that describes that window. This process is completed for each window. The result is a feature descriptor that is a normalised 64-variable vector.

The OpenCV library provides two types of SURF algorithms: the normal SURF method that was just discussed and a rotation variant version called U-SURF (or Upright SURF). U-SURF provides faster feature detection in return for SURF's rotation invariance. For the AR Campus Tour application, it is better to enable users to take photos at whatever angle they prefer. Thus, only the normal SURF algorithm was tested.

### 2.3.3 ORB (Orientated FAST and Rotated BRIEF)

The ORB algorithm is divided into the Orientated FAST algorithm for feature detection and the Rotated BRIEF algorithm for feature description.

**FAST** The FAST (or Features from Accelerated Segment Test) was developed to be used in real-time applications. Proposed by Rosten and Drummond [36], it uses machine learning algorithm to perform speedy corner detection. The algorithm works as follows:

- Select a pixel  $P$  in the input image as a potential feature. The intensity of this pixel is  $I_p$ .
- Get the intensity of the 16 pixels that create a circle around  $P$ , as shown in Figure 2.12. Call these intensities  $I_1$  to  $I_{16}$ .
- Select an appropriate threshold value  $T$ .
- $P$  is a corner if there exists a set of 12 contiguous pixels, where the intensity of all pixels is brighter than  $I_p + T$  or the intensity of all pixels is darker than  $I_p - T$ .
- After some iterations the algorithm generates a decision tree classifier, so that feature detection can be sped up.

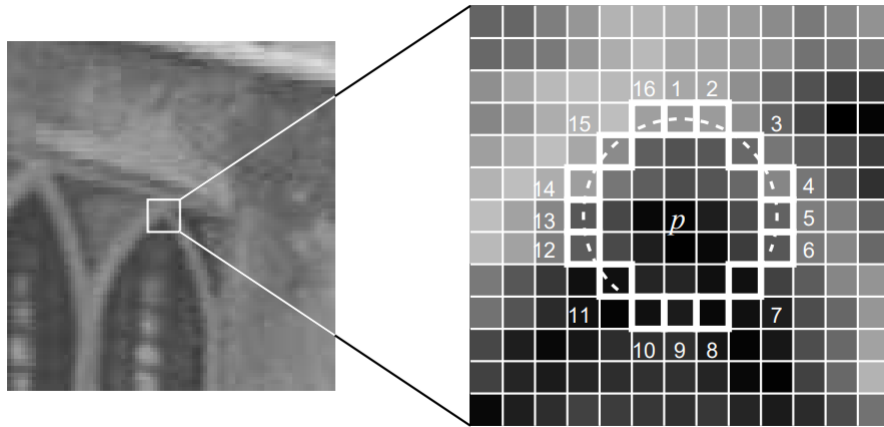


Figure 2.12: FAST Corner Detection [36]

FAST is a quicker algorithm than SIFT however there are some disadvantages to using FAST. The algorithm is not robust against images with high levels of noise and is dependent on the threshold value  $T$ . Picking a value for  $T$  that is too high or too low will result in poorly performing algorithm. Although the speed-up is a desirable feature of the algorithm, this could cause difficulties in a tourism application. The value for  $T$  was chosen to be 10, as recommended by OpenCV.

FAST is not rotation-invariant, meaning that the algorithm performs poorly with rotated images. However, it was modified to calculate the orientation for ORB. This version is called Orientated FAST. After a pixel is determined to be a corner, a weighed centroid is calculated based on the intensity of the circling pixels (i.e.  $I_1$  to  $I_{16}$ ). The orientation of the feature is represented as the direction of the vector from the centre pixel to the centroid.

**BRIEF** BRIEF (Binary Robust Independent Elementary Features) [37] is a feature description generator that avoids using large vectors as descriptors like in SIFT to reduce the memory and time required to match features. It produces the description by taking some region around the keypoint. Within this region, a number of pixel pairs are selected (usually 128, 256 or 512 pairs) and their intensity values are compared. If the first pixel has a greater intensity, then a value of '1' is assigned and '0' otherwise. This is done for all pixel pairs and the result is a string of bits. The length of this string is equal to the number of pixel pairs selected. This is why numbers that are



multiples of 8 such as 128, 256 or 512 are selected. The pixel pairs can be selected in multiple ways according to [37]. One example, is to randomly and uniformly choose  $x$  and  $y$  values for the pixels to be compared.

BRIEF is also used to match the features it describes. Using binary strings instead of vectors of floating-point values provides a speed-up to the feature matching stage by using Hamming distance. This is a simple operation, that only requires a XOR operation on the two feature descriptors being compared, followed by a simple bit count operation.

**ORB** ORB (or Oriented FAST and Rotated BRIEF) [38] is designed to be an efficient alternative to SIFT by combining algorithms that have previously been mentioned. ORB uses the FAST algorithm for feature detection and the BRIEF method of creating feature descriptors. The algorithm also includes the application of Harris Corner Detection [39] to measure the keypoints and determine which are best for the feature matching step.

## 2.4 Feature Matching

Once the feature detection stage has been completed, feature matching must be implemented. Similar to the feature detection algorithms, there are multiple feature matching algorithms. The following section will compare two different methods of feature matching: Brute-Force Matching and FLANN-Based Matching. Note that these feature matchers are only compatible with SIFT and SURF because they expect feature descriptors to be vectors of floating point numbers. ORB uses bit strings as descriptors and therefore cannot be implemented with the matchers described in this section. Instead ORB uses the BRIEF algorithm to carry out feature matching.

### 2.4.1 Brute-Force Matching

A brute-force matcher is as simple as the name suggests. The matcher takes each of the descriptors from the identified features in the first image and matches it with the descriptors in the second image. To identify a match, one feature in the first image is compared to all the features in the second image using some distance formula. The

feature in the second image that returns the closest distance is returned as the match. There are different distance formulae depending on which matching method is being used. As the descriptors are floating-point values, the normalised L1 or normalised L2 distance formulae are suitable (normalised L2 distance is preferred). There is an optional cross check matcher that performs the matching process twice: once going from the first image to the second image, and once going from the second image to the first. It only returns the feature pairs that are the best match in both tests. This sacrifices speed in return for reliability.

### **2.4.2 FLANN-Based Matching**

FLANN is a Fast Library for Approximate Nearest Neighbours, a collection of highly optimised algorithms for fast nearest neighbour operations in large databases or high dimensional features. This library can be used to implement a nearest-neighbours matcher, which is similar to a Brute-Force matcher, but it returns the top k matches, where k is decided by the developer. A FLANN-based matcher performs better than a brute-force matching for large datasets, e.g. a large image.

# Chapter 3

## Design

### 3.1 Pipeline

The pipeline for this application is divided into two main sections (Figure 3.1). The first part consists of the functionality that occurs on the user's mobile device. The second part includes all of the online functionality. This design is similar to other AR Image Recognition applications [11], [29].

The user's mobile device has the ability to take images and also capture information related to the image which will assist with image recognition later on in the pipeline. It captures the GPS location and camera pose of the user's device when the image was taken, which helps narrow down the search for which landmark is pictured. The device then uploads the image and the appropriate information to the Image Recognition Server. Finally, the device receives a result from the Image Recognition Server.

The online section of the pipeline is made up of a database and an Image Recognition Server. The online database stores all of the reference images of landmarks with the GPS and pose information associated with them. The Image Recognition Server will take the user's image and attempt to match it to one of the images stored in the database. As mentioned previously, the additional information that is uploaded by the user is used to reduce the time of finding the correct landmark. The Image Recognition Server will return a result to the user's device indicating if the matching was successful or not. If the match is successful, the server will return additional information.

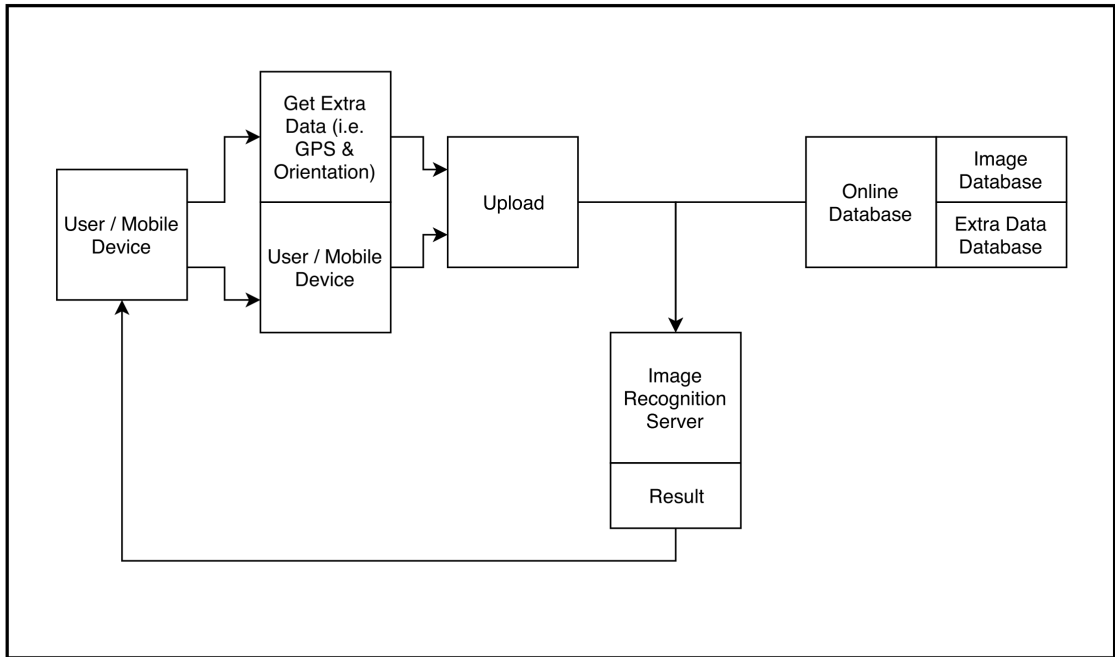


Figure 3.1: Pipeline

## 3.2 Mobile Device System

The only actor involved in this system is the user of the mobile device (Figure 3.2). The main functionality of this system is to upload the image and corresponding information. The upload image function uses the device's camera to take a picture. To upload the additional information, the data is retrieved from the device's gyroscope and the GPS system. Finally, the device must be able to accept and handle the data that is returned from the image recognition server.

## 3.3 Database System

The Database Systems use case diagram is very straightforward (Figure 3.3). The two actors involved are the image database handler and the extra information database handler. Both of them must have the ability to save information of the appropriate type into the database. Additionally, they must also have the ability to retrieve the information so that it can be used by the image recognition server.

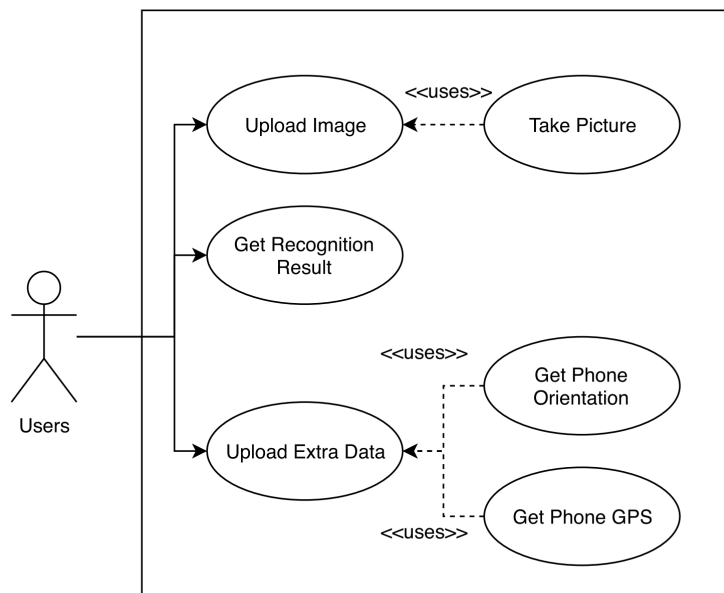


Figure 3.2: Mobile Device System



Figure 3.3: Database System

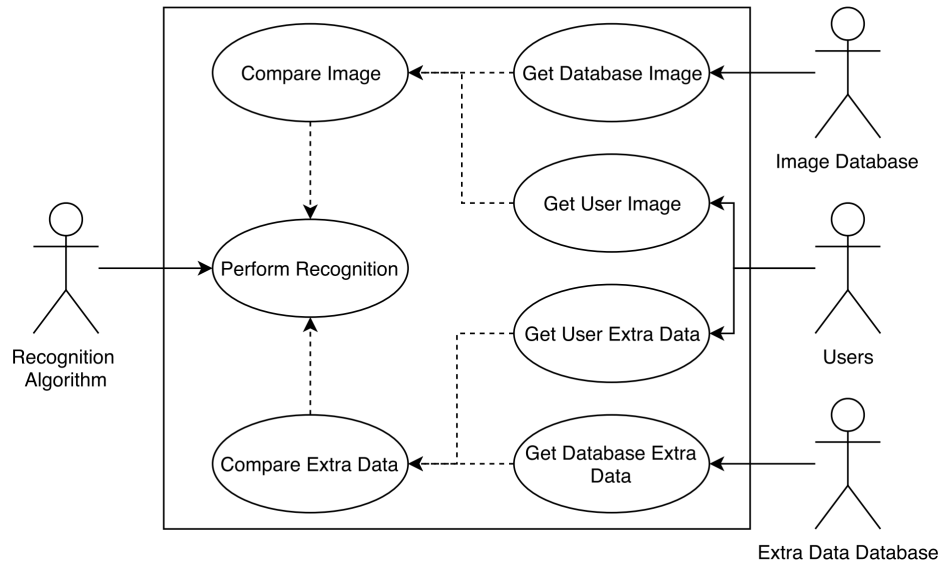


Figure 3.4: Recognition System

### 3.4 Image Recognition System

The Image Recognition System is the most complex system in the application. It is the core system of the application and interacts with all of the other systems. To perform an image recognition test, the system needs to retrieve the device’s image and extra information and compares it to images and information stored in the database. This has to be retrieved by the two database handlers.

# Chapter 4

## Implementation

### 4.1 Technologies Used

**Android** Android is a Linux-based operating system used in mobile devices. As of 2018, Android accounts for 86% of the mobile operating system market share, which is over 2 billion mobile devices [40]. When an application is developed for the Android platform it is able to run on most other Android devices (as long as the Android version is compatible). This app uses Google ARCore which requires Android OS with API level 24 (also known as Android Version 7.0 or Nougat). Additionally, the mobile device requires access to the internet and an in-built camera.

**Google ARCore** ARCore is a AR platform developed by Google. It provides the functionality necessary for building AR applications including motion tracking of a device relative to the world, environmental understanding with flat surface detection and objection detection, and light estimation of the environment [41]. The AR Campus Tour application uses ARCore as the base for AR functionality.

**Mapbox** Mapbox is a platform that provides location data for applications. The provided functions include map generation, location searching and navigation [42]. Mapbox also provides an AR version of their platform. This provides a library that is built upon ARCore's Unity API, so all of Mapbox's services can be used in an AR application.

**Unity Engine** Unity is video game physics engine that is used in many video games across multiple platforms. The Unity Editor is a piece of development software that is used to create games that run on the Unity Engine [43]. Unity was used to build the AR Campus Tour application because both ARCore and Mapbox both have APIs that integrate well with the Unity Engine. Unity also links with the Android Development Studio to build projects on Android devices.

**OpenCV** Open-CV is the Open Source Computer Vision Library. It provides an extensive range of computer vision and machine learning algorithms [44]. The algorithms of interest are the object identification algorithms, which are used in the server to identify landmarks in the query images.

**Flask** Flask is a micro-framework written in Python for building websites and web applications. It is used in this project as the base to the web server [45]. It handles all the requests that the user's device sends while using AR Campus Tour. This includes both API requests and image recognition requests.

**JSON** JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easily read and written by humans but also easy for machines to parse. It follows the conventions of the C-family of programming languages (e.g. C, C++, C#, Java, Python) [46]. JSON is used in the AR Campus Tour application to exchange information between the user's device and server, such as tour information and attraction information.

## 4.2 Technical Architecture

Figure 4.1 demonstrates how each of the technologies and tools used in this project interact with each other. Unity is used to create the application and build it on Android devices. Within Unity, ARCore and Mapbox APIs provide AR functionality and location-based data and objects to the application. The user must have an Android device with the correct Android version installed to run the app. The server was built using Flask and Python. The application makes requests to the server which Flask



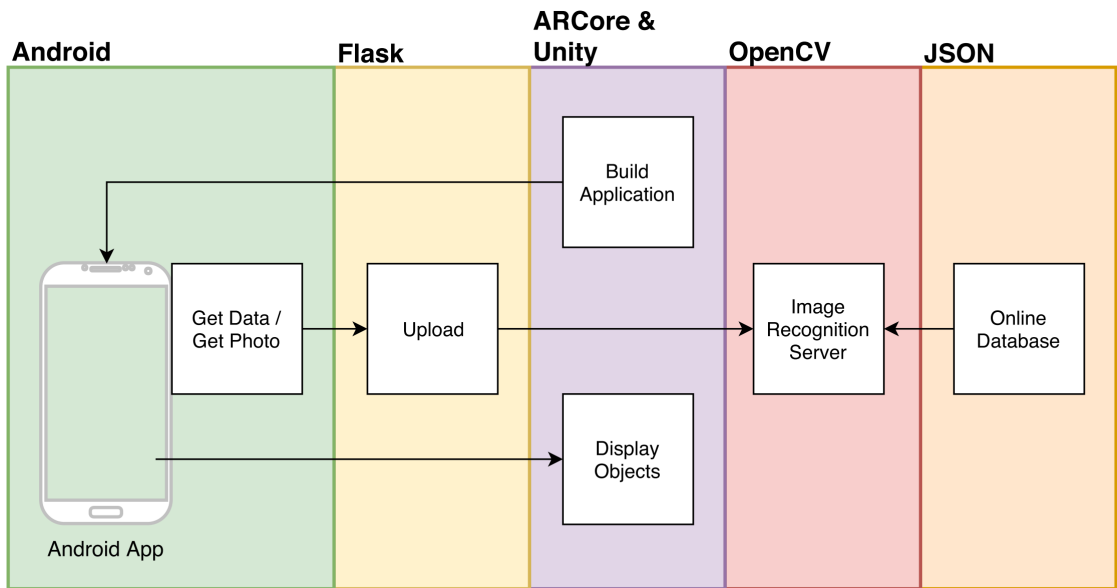


Figure 4.1: Technical Architecture Diagram

handles. Running within the server is the OpenCV library, which carries out the recognition process. Data is sent to and from the server in JSON format.

### 4.3 UI/UX Design

This section will list and explain the different UI/UX elements present in the AR Campus Tour Application. Figure 4.2 shows all the major UI/UX elements which are labelled with red numbers.

The Capture Button (labelled 1) activates the image search and recognition process. When the user presses this button, the camera's output (or the background to the app) is captured, ignoring all the UI elements that may prevent the recognition algorithm from achieving an accurate result. If the recognition process is successful than an AR Landmark Marker (labelled 5) is created. This object has the name and a piece of information regarding the identified landmark. If the recognition is unsuccessful a Nothing Marker (labelled 6) is created.

The Tour Selection Menu (labelled 2) enables the user to select which tour they want to do. On start up, this menu will only have a loading message displayed. This indicates that the application is requesting the data required to initialise a tour from

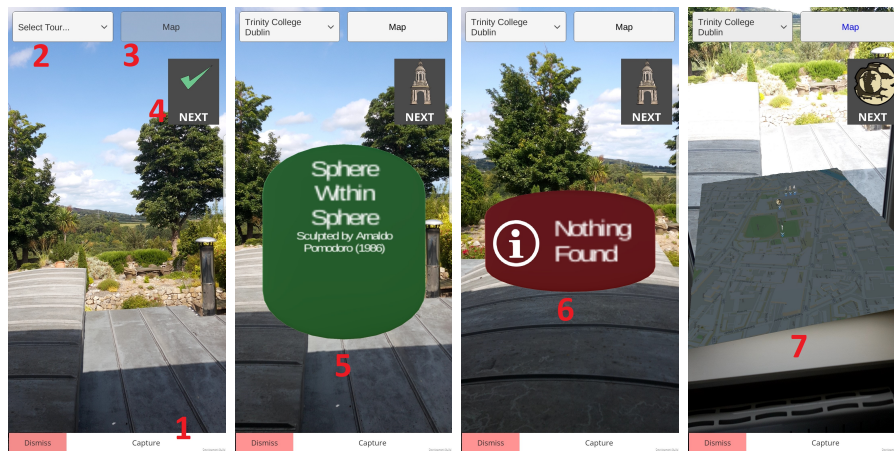


Figure 4.2: UI Elements of AR Campus Tour

the server. The response to this request includes the basic data regarding all the tours. With this information, the Tour Selection Menu is filled with the names of each of the tours. Now the menu behaves as a dropdown menu and the user can select a tour.

The Map Button (labelled 3) toggles the AR Map (labelled 7). The Map Button is inactive until the user selects a tour from the Tour Selection Menu. Activating the AR Map will produce and spawn an AR object in front of the device's camera. The AR object is a 3D map of the area the tour takes place in. The map also features markers showing the position of points of interest and a 3D marker showing the location of the user and the direction they are facing. Pressing the Map Button again will deactivate the AR Map.

The Next Landmark Icon (labelled 4) indicates to the user what the next expected landmark is. It is initialized when the user selects a tour from the Tour Selection Menu, and updates when the user identifies the next expected landmark.

## 4.4 Application Walkthrough

This section maps out how a user would typically use the AR Campus Tour application. The accompanying figures are in the Appendix - Figures 1 to 8.

**Initial steps** Once the app has started up (Figure 1), an API request is sent to the server asking for information regarding tours. This fills the Tour Select Menu with

the names of the tours, enabling the user to select which tour they want to begin (Figure 2). By clicking the Tour Select Menu, a dropdown menu expands listing all the tours currently available to the user. The user selects the desired tour (Figure 3). Another API request is made, requesting information regarding only that tour. The server returns a list of all the attractions in the tour, along with the GPS positions of those attractions. With this information a Tour Progress Manager object is created that stores Array Lists of the landmark names, the GPS coordinates and Boolean values marking whether the user has visited the landmark. This object tracks the landmarks that the user has visited and keeps track of the users progress of the tour. Using the Tour Progress Manager, the Next Landmark Icon is changed to represent the first landmark on the tour (Figure 4). Now, the user can begin to walk around with their device and take images. The user walks to a landmark, holds their device up to the landmark and presses the Capture Button. This initializes the image recognition process.

**Server Behaviour** The device sends the captured image, the GPS information and camera pose to the server. With this data, the server first determines which of the landmarks the user could be looking at based on the GPS location and removes the landmarks that are too far away from the user. The next step is to narrow down the number of images that need to be searched. This is done using the camera pose information. Images that have similar camera poses to the query image will be prioritised to try and find a good match in the shortest amount of time. Once a list of likely images has been found, the matching algorithm can begin. The features of the query image and likely images are found using feature detection. The query images features are compared with each of the likely images features using feature matching. The result of the feature matching is a score that represents the similarity between two images. A score of 0 is a perfect score and means the two images are a complete match. The higher the score the less of a match the two images are. This score is calculated for each feature pair. The server sorts the matches by score, from lowest to highest and takes the sum of the top ten matches and uses that as the final score of the match between the two images. This process is repeated between the query image and each of the likely images. At the end of the process, the image with the lowest score, and that score being below the threshold, is now considered the landmark being looked at.

The name and information associated with this image is returned to the user's device in the form of a JSON response. The JSON response includes the landmark name, the display name of the landmark (the name the user sees), the information regarding the point of interest and the position on the device's screen that marks the centre of the landmark. However, there are two cases where the server will return a negative result, i.e. a landmark was not found. Firstly, if the position of the user is too far away from any landmarks, a negative result is returned before any image recognition is performed. Secondly, if all the images return a score that is above the threshold, no adequate match is found, and a negative result is returned. This score threshold is decided by the developer. In these cases, the JSON response simply returns the string None.

**Processing Server Result** Based on the response, the application generates one of two AR objects. For a negative response, the application produces a Nothing Marker. This red marker is positioned in the environment directly in front of the device's camera. It has a notification icon, as well as the text Nothing Found, indicating to the user that the landmark search was unsuccessful (Figure 5). This marker disappears 10 seconds after its creation, so that the user's display is overrun by these markers. In the case of a positive response, an AR Landmark Marker is created. This marker holds the display name received from the server at the top and a small piece of information printed below. Similar to the Nothing Marker, this object is spawned directly in front of the device's camera. If the camera is moved horizontally or vertically, the AR Landmark Marker retains its position in the augmented environment. On a successful result, the Tour Progress Manager is updated, and the identified landmark is marked as visited. If the landmark identified is the next expected landmark, the Next Landmark Icon is updated based on which landmark the Tour Progress Manager returns as the next landmark (Figure 6).

**Using the AR Map** By pressing the Map Button in the top-right corner of the screen, the AR Map spawns in front of the device's camera. This is a 3D map of the area in which the tour is taking place. In the case of this app, the map is centred on Trinity College Dublin (Figure 7). There are 2D markers that show the positions of the landmarks. These markers consist of a tower of three icons. In the centre, there

is an icon that represents the landmark. The icon is a cartoon drawn version of the landmark to guide users towards the correct points of interest. The bottom icon is a downward pointing arrow that points to the point on the 3D where that landmark is found. The final icon on the top is a tick icon that indicates whether the user has identified the landmark. If the user has yet to visit the landmark, the icon is invisible and appears when the user identifies the landmark through the app (Figure 8). Pushing the Map Button again will deactivate the AR Map and remove it from the augmented environment.

# Chapter 5

## Results

The three algorithms mentioned in Section 2.3 were tested for their efficiency. The algorithms' accuracy and speed were compared against each other. For an application like the AR Campus, the ideal algorithm should have 100% accuracy and correctly identify landmarks every time with minimum time taken to perform the image matching. To conduct this test, five landmarks were chosen in Trinity College Dublin. Multiple images of these landmarks were taken and cropped so that there was as little of the background as possible seen in the images. This reduces the amount of features identified that were not on the landmark. These features are noise and would disrupt the algorithm tests. These cropped images were placed in the database and are used by the server during the image recognition process. Finally, one more image was taken for each landmark. These images remained uncropped and were used as the query images. The database and query images can be seen in Appendices 9 to 14.

To test the accuracy of the algorithms, each query image was matched against the database images. The test will have a positive result if the match with the lowest score is between the query image and a database image of the same landmark as in the query image. Otherwise, it is a negative result. To measure the speed of the algorithm, the time taken to complete the image recognition process is measured. The timer starts just before the query image is loaded and ends when results of the final match is found. The results of the image recognition algorithms test is layout in Table 5.1.

The SIFT algorithm performed well. It successfully identified all five landmarks with both matchers. However, on average, SIFT (BF) took just over two minutes to

| Algorithm | Accuracy | Average Time     | Minimum Time    | Maximum Time     |
|-----------|----------|------------------|-----------------|------------------|
| SIFT (BF) | 100%     | 123.15 sec/Image | 19.90 sec/Image | 392.46 sec/Image |
| SIFT (FL) | 100%     | 101.09 sec/Image | 17.48 sec/Image | 294.95 sec/Image |
| SURF (BF) | 100%     | 101.00 sec/Image | 16.64 sec/Image | 313.07 sec/Image |
| SURF (FL) | 60%      | 54.80 sec/Image  | 16.75 sec/Image | 135.02 sec/Image |
| ORB       | 100%     | 0.48 sec/Image   | 0.34 sec/Image  | 0.69 sec/Image   |

Table 5.1: Algorithm testing results - BF means using the Brute Force feature matcher, FL means using the FLANN matcher

complete one match (i.e. feature detection of the database image and feature matching between query image and the database image). SIFT (FL) performed slightly better with an average time of just over a minute and a half. The AR Campus Tour application will most likely search through multiple images. Making the user wait for minutes before receiving a result is unacceptable for this sort of application. Moreover, the range of time taken is varied. The shortest time recorded for matching two images is 17.48 seconds (FL) and the longest time recorded is 392.46 seconds (BF). This variance in time is due to the number of features detected in the images.

The SURF algorithm has extra estimation steps during the feature detection steps, thus the accuracy begins to suffer when used with the FLANN matcher. SURF (FL) misidentified two out of the five landmarks. However, because of the extra estimation steps, SURF is able to quickly eliminate any impossible matches. It does this by determining if a feature is white on a black background or black on a white background. It eliminates potential matches if the features are different colours. Thus, there is a significant speed-up when compared to SIFT. Using SURF with the BF matcher performs better on accuracy, but loses the big time advantage over either of the SIFT algorithms. On average SURF (FL) takes 54.80 seconds per image and SURF (BF) takes 101 seconds. Although this is faster than both SIFTs, it is still an unacceptable time for the AR Campus Tour application.

When comparing the Brute Force and FLANN feature matchers, FLANN completed the feature matching quicker than its Brute Force counterpart. This is because FLANN performs better than Brute Force only on very large images. However, when using the FLANN matcher in combination with SURF, the algorithm’s accuracy fell to 60%. In the context of AR Campus Tour, this translates into the misidentification of

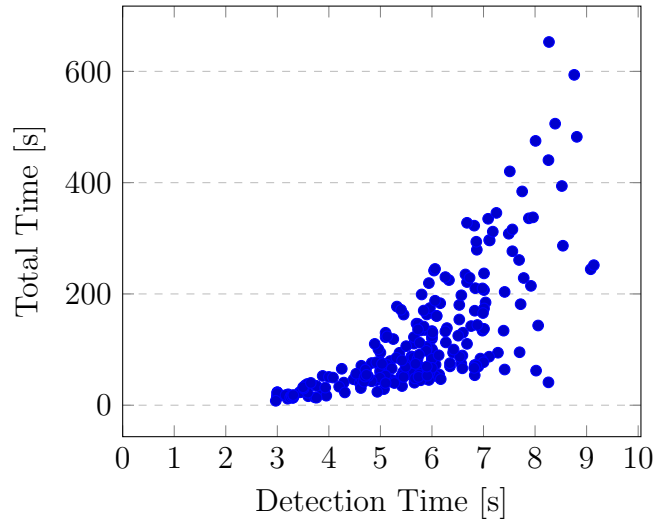
landmarks. Therefore, if choosing between SURF and SIFT, SIFT with the FLANN matcher provides good accuracy in the shortest time.

The ORB algorithm's combination of using the FAST feature detection and BRIEF feature matching achieves a high accuracy and low time. The algorithm successfully identified all five landmarks, and the average time taken to perform a match is a fraction of the average time of SIFT or SURF. The main reason for this is the feature descriptors in ORB are byte strings. Descriptors in SIFT and SURF are vectors of floating-point numbers. This means that the difference between features is much quicker to calculate. The ORB algorithm provides a good, usable time for the AR Campus Tour application. However, during the image recognition process, if multiple database images are matched, the user will be left waiting for their device to receive a response for too long. This is why we include the GPS and camera pose data to reduce the time taken to recognise landmarks. Regardless, out of the algorithms tested, ORB is the ideal algorithm for tourism applications.

Further investigation into the algorithms' time was carried out. Firstly, the ratio of the times taken during feature detection and feature matching was obtained. The results are shown in Figure 5.1. It shows that when the overall process takes a relatively short amount of time, the feature detection process takes up a small percentage of the total time of the process. However, as the process becomes more complex and takes more time, the harder it is to predict how long the feature detection will take. Nevertheless, the feature detection always accounted for the quicker part of the image recognition process. On average, time spent on feature detection was only 8.5%, with a lower bound 1.27% and an upper bound of 37.45% of the overall time. This shows that by focusing efforts on speeding up the feature matching stage alone, a significant speed up to the overall process can be achieved.

Following this, the feature matching process was examined further. In particular, the effect of the number of features identified on the time of the overall process was investigated. The expected result was that the higher number of features detected, the more feature matching and therefore, the more time it takes. The results of this test are laid out in Figure 5.2. This is a comparison of the total number of features found in both the query and database image against the time taken to perform the image recognition process. The graph shows that the time taken is highly dependent on the number of features found in the query image, as highlighted in Figure 5.3. However,





*Figure 5.1: Feature detection time against complete image recognition process time*

if the query image is constant during all the tests, a linear relationship between the number of features and time taken is observed (Figure 5.4). When a query image with a low number of features detected is used, the y-intercept of the graph is a low value of time (i.e. the minimum time taken to complete feature matching is low) and the slope of the graph is quite flat (i.e. an increase in the number of features detected in the database image will not have a large effect on the overall time taken). However, having an image with a large number of features will result in a higher minimum time to complete the process and a steeper slope, so an increase in the number of features in the database image will have a bigger increase in the overall time taken. Therefore, by reducing the number of features in the images, particularly in the query image, then a speed up will be achieved. However, there is a risk. By reducing the number of features detected, will also reduce the accuracy of the image recognition algorithm.

One of the reasons why the ORB algorithm is very fast when compared to SIFT or SURF is that it limits the number of keypoints it finds to 500. SIFT and SURF can find thousands of features per image. However, a limit can be applied to SIFT so that it detects a maximum of 500 features. SURF can be modified to detect less features by increasing a parameter called the hessian threshold. The hessian threshold was increased from 100 to 2000. New tests were carried out with SURF and SIFT to see how these modifications effect their accuracy and speed. The results are shown in

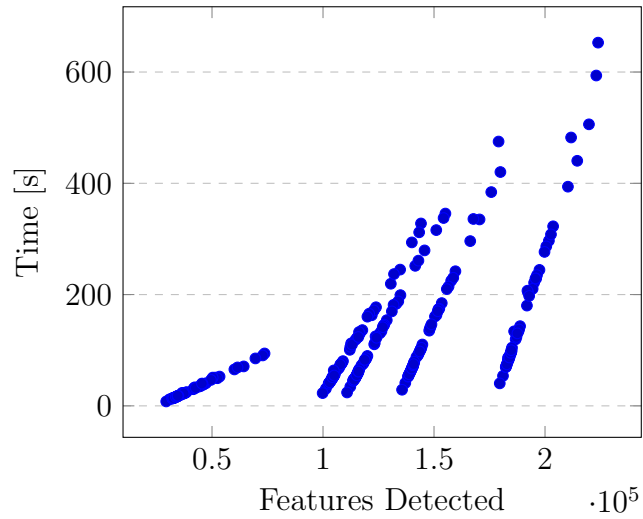


Figure 5.2: Number of Features Detected against Time Taken to perform image recognition

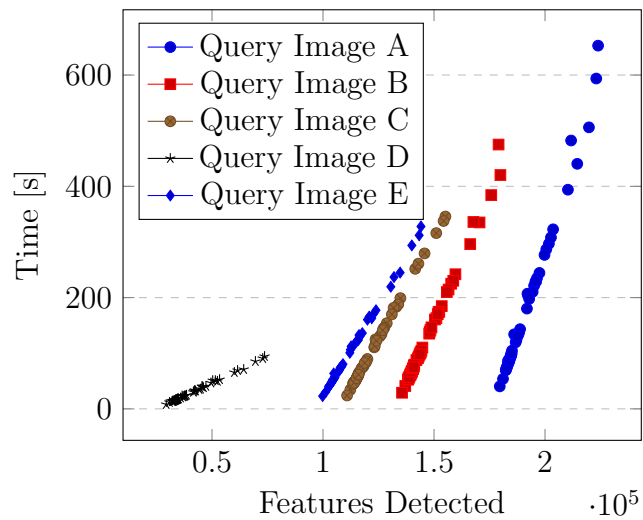


Figure 5.3: Number of Features Detected against Time Taken to perform image recognition separated by query image

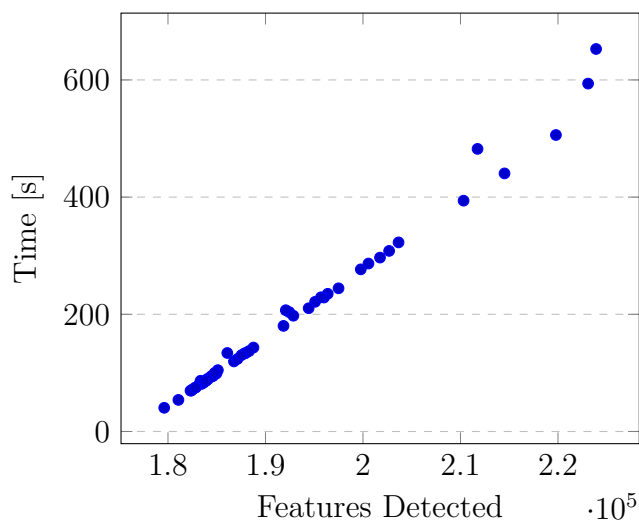


Figure 5.4: Number of Features Detected against Time Taken to perform image recognition for one query image

| Algorithm       | Accuracy | Average Time   | Minimum Time   | Maximum Time   |
|-----------------|----------|----------------|----------------|----------------|
| SIFT (BF, 500)  | 100%     | 3.52 sec/Image | 2.85 sec/Image | 4.66 sec/Image |
| SURF (BF, 2000) | 100%     | 3.61 sec/Image | 2.45 sec/Img   | 6.40 sec/Image |

Table 5.2: Algorithm testing results - SIFT is limited to 500 features and SURF has a hessian threshold of 2000

Table 5.2.

Limiting the number of features detected results in a massive increase in speed for both algorithms. More interestingly, both achieved an accuracy of 100%. When finding the final score for images, the scores of the best 10 matching features are taken. Therefore, it is not necessary to use tens of thousands of features (as was the case in the original algorithm tests). 500 features in both the query image and database image is sufficient to achieve an accurate result. Another interesting result to note is the ratio of feature detection time to feature matching time. The feature matching stage is now the quicker part of the image recognition process and accounts for a small percentage of the total time taken, as shown in Figures 5.5 and 5.6. However, the image recognition process will still take a few seconds per image due to the feature detection stage, and this may be too slow for users (considering that multiple images will be used during the image recognition process). There are methods that can optimise feature detection

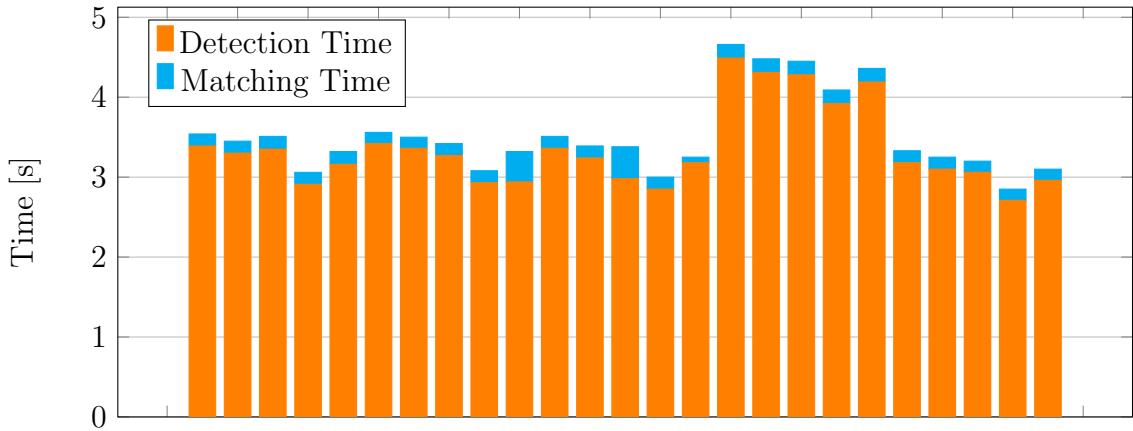


Figure 5.5: Features detection time against feature matching time in SIFT (BF, 500)

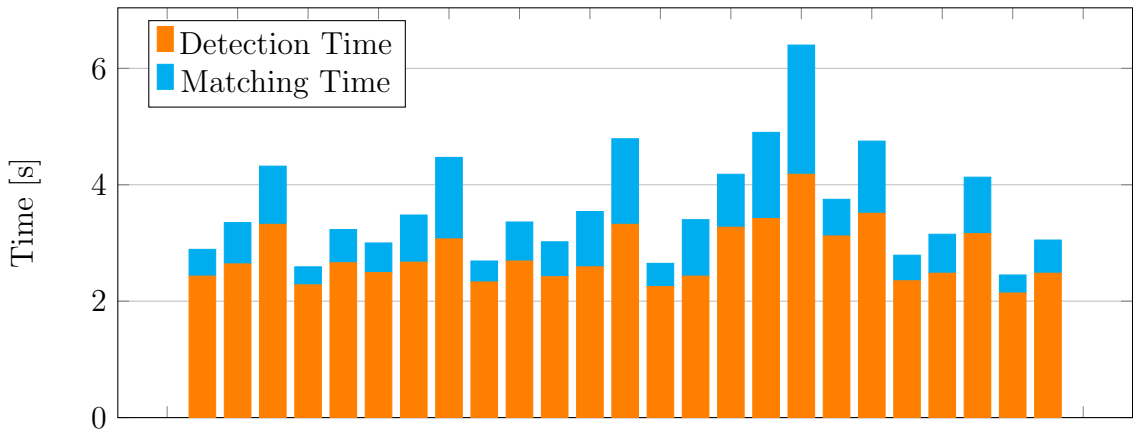
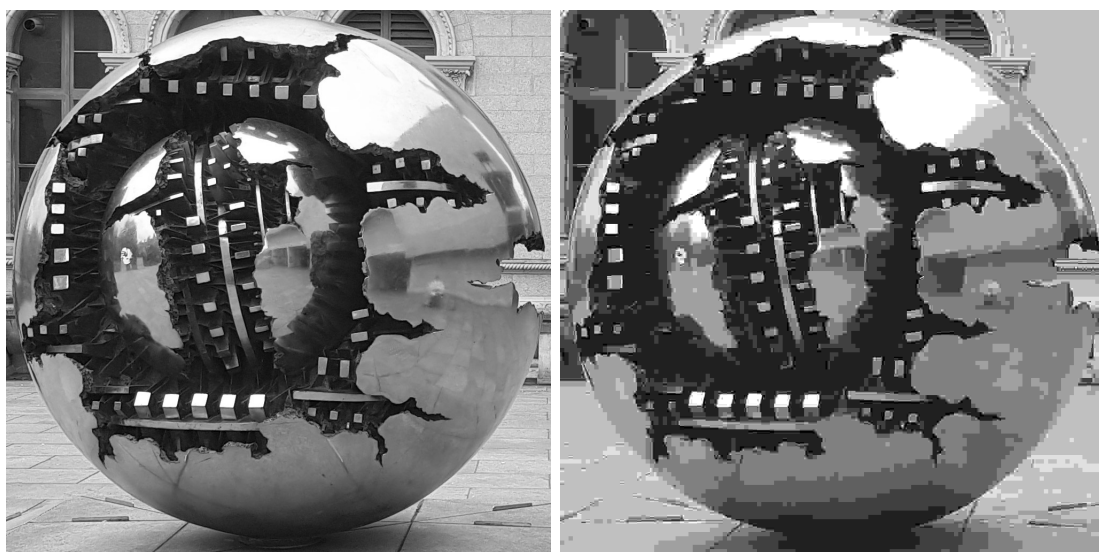


Figure 5.6: Features detection time against feature matching time in SURF (BF, 2000)

time. For example, saving the feature descriptors of images in a file, and then reading the file instead of generating a new set of descriptors each time the algorithm is run.

Another method of reducing time is to use image compression or lower the image quality. The SIFT algorithm was tested again using the same query images and database images. However, when the images were loaded, the quality of the image was lowered to 10% of the image’s original quality (Figure 5.7). This was done using OpenCV’s compression algorithms. The expected result was to see an increase in the speed of the image recognition process but see a decrease in the server’s ability to distinguish between landmarks. The image quality was reduced to a low amount (10%) to try and achieve a significant speed-up when compared to no compression. On average,



*Figure 5.7: Normal image on the left, compressed image on the right*

there was a reduction in only 5% in the number of features detected, which only results in a 10% speed up. However, this version of the algorithm misidentified two out of the five landmarks.

# Chapter 6

## Conclusions

This paper has shown that it is possible to create a tourist AR application in the form of AR Campus Tour. All the fundamental requirements for such an app are present including landmark recognition, information display and mapping navigation and guidance. This paper also describes a design for such applications that others can follow to create their own applications and build on the research in this area. The technologies that were used to build the AR Campus Tour application are listed and have been shown to work well in the context of a tourism application. Finally, this paper delved into the different possible image recognition algorithms that could be used and concluded that an algorithm that has a low feature matching time but high recognition accuracy is best suited for a tourism application, as well as modification that can be made to help speed up this process. This paper finds ORB by [38] to be an idea algorithm.

# Chapter 7

## Further Research

**AR Object Interaction** The AR Campus Tour application produces an AR Landmark Marker object whenever a landmark is successfully identified. However, this object only displays the name of the landmark identified as well as some additional information about it. There is potential to create an AR Landmark Marker object that is interactable, similar to the CorfuAR application [17]. For example, the user could tap this object when it is visible on screen to access additional information, as well as media such as photos, videos and audio files. Including audio cues and information about each landmark is shown to increase tourists' willingness to pay more at a tour site [24].

**Accessibility** One potential issue with the AR Campus Tour application is that it is reliant on WiFi or mobile network connectivity. All information about tours and landmarks are stored on the server. The server can only be accessed by the application if the user's device has a connection to the Internet. This means that areas with poor WiFi and network coverage, as well as data-roaming charges, must be considered. A possible solution is to allow users to pre-download data so that they can experience the tour undisturbed in an offline mode.

**User Co-creation of Content** User co-creation of content is a key component in making a good technology integrated tourism application [8]. This is a feature that the AR Campus Tour application has yet to implement. Enabling users to post images taken with the application on social media is a good way for users to share their

experience and attract more visitors to the tour site [10]. Alternatively, users could be given the ability to create custom tours by linking their preferred landmarks [16] and share these custom tours with others.

**Deep Learning and Neural Networks** A subject in the area of Machine Learning that is currently being researched is Deep Learning. Convolutional Neural Networks are networks that implement Deep Learning to complete highly complex image recognition problems. Networks such as AlexNet [47] and ResNet [48] can be used instead of SIFT/SURF/ORB in this project. Investigating the performance of these networks would further enhance this project.

**Hardware** The algorithms were tested on one machine only. However, different machines will have different CPU, GPU and RAM capabilities which will have different effects on the performance. Extending the comparison of algorithms to include different hardware would help get closer to finding the ideal design for a tourist AR application.



# Bibliography

- [1] D. Buhalis, “Strategic use of information technologies in the tourism industry,” *Tourism management*, vol. 19, no. 5, pp. 409–421, 1998.
- [2] B. J. Pine, J. Pine, and J. H. Gilmore, *The experience economy: work is theatre & every business a stage*. Harvard Business Press, 1999.
- [3] E. Binkhorst and T. Den Dekker, “Agenda for co-creation tourism experience research,” *Journal of Hospitality Marketing & Management*, vol. 18, no. 2-3, pp. 311–327, 2009.
- [4] Y. Wang and D. R. Fesenmaier, “Towards understanding members’ general participation in and active contribution to an online travel community,” *Tourism management*, vol. 25, no. 6, pp. 709–722, 2004.
- [5] J. Fotis, D. Buhalis, and N. Rossides, “Social media impact on holiday travel planning: The case of the russian and the fsu markets,” *International Journal of Online Marketing (IJOM)*, vol. 1, no. 4, pp. 1–19, 2011.
- [6] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, “Smart tourism: Foundations and developments,” *Electronic Markets*, vol. 25, no. 3, pp. 179–188, 2015.
- [7] D. Buhalis and A. Amaranggana, “Smart tourism destinations enhancing tourism experience through personalisation of services,” in *Information and communication technologies in tourism 2015*, Springer, 2015, pp. 377–389.
- [8] B. Neuhofer, D. Buhalis, and A. Ladkin, “A typology of technology-enhanced tourism experiences,” *International Journal of Tourism Research*, vol. 16, no. 4, pp. 340–350, 2014.
- [9] C. K. Prahalad and V. Ramaswamy, “Co-creation experiences: The next practice in value creation,” *Journal of interactive marketing*, vol. 18, no. 3, pp. 5–14, 2004.

- [10] I. P. Tussyadiah and D. R. Fesenmaier, "Mediating tourist experiences: Access to places via shared videos," *Annals of Tourism Research*, vol. 36, no. 1, pp. 24–40, 2009.
- [11] B. Zhang, "Design of mobile augmented reality game based on image recognition," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 90, 2017.
- [12] W. Chen, "Historical oslo on a handheld device—a mobile augmented reality application," 2014.
- [13] T. Jung, M. C. tom Dieck, H. Lee, and N. Chung, "Effects of virtual reality and augmented reality on visitor experiences in museum," in *Information and Communication Technologies in Tourism 2016*, Springer, 2016, pp. 621–635.
- [14] Z. Yovcheva, D. Buhalis, and C. Gatzidis, "Engineering augmented tourism experiences," in *Information and communication technologies in tourism 2013*, Springer, 2013, pp. 24–35.
- [15] D. Cianciarulo, "From local traditions to ffdfffdfffd augmented reality ffdfffdfffd. the muvig museum of viggiano (italy)," *Procedia-Social and Behavioral Sciences*, vol. 188, pp. 138–143, 2015.
- [16] M. Umlauft, G. Pospischil, G. Niklfeld, and E. Michlmayr, "Lol@, a mobile tourist guide for umts," *Information Technology & Tourism*, vol. 5, no. 3, pp. 151–164, 2002.
- [17] P. E. Kourouthanassis, C. Boletis, and G. Lekakos, "Demystifying the design of mobile augmented reality applications," *Multimedia Tools and Applications*, vol. 74, no. 3, pp. 1045–1066, 2015.
- [18] T. Höllerer and S. Feiner, "Mobile augmented reality," *Telegeoinformatics: Location-Based Computing and Services. Taylor and Francis Books Ltd., London, UK*, vol. 21, p. 00 533, 2004.
- [19] M. T. Linaza, D. Marimón, P. Carrasco, R. Álvarez, J. Montesa, S. R. Aguilar, G. Diez, *et al.*, *Evaluation of mobile augmented reality applications for tourism destinations*. na, 2012.
- [20] J. Short, E. Williams, and B. Christie, "The social psychology of telecommunications," 1976.

- [21] M. Kang and U. Gretzel, “Effects of podcast tours on tourist experiences in a national park,” *Tourism Management*, vol. 33, no. 2, pp. 440–455, 2012.
- [22] K. M. Lee, *Social responses to synthesized speech: Theory and application*. Stanford University, 2002.
- [23] E. Cranmer, T. Jung, A. Miller, *et al.*, “Implementing augmented reality to increase tourist attraction sustainability,” 2016.
- [24] Z. He, L. Wu, and X. R. Li, “When art meets tech: The role of augmented reality in enhancing museum experiences and purchase intentions,” *Tourism Management*, vol. 68, pp. 127–139, 2018.
- [25] A. L. Kečkeš and I. Tomičić, “Augmented reality in tourism—research and applications overview,” *Interdisciplinary Description of Complex Systems: INDECS*, vol. 15, no. 2, pp. 157–167, 2017.
- [26] C. D. Kounavis, A. E. Kasimati, and E. D. Zamani, “Enhancing the tourism experience through mobile augmented reality: Challenges and prospects,” *International Journal of Engineering Business Management*, vol. 4, p. 10, 2012.
- [27] R. Fatima, I. Zarrin, M. A. Qadeer, and M. S. Umar, “Mobile travel guide using image recognition and gps/geo tagging: A smart way to travel,” in *Wireless and Optical Communications Networks (WOCN), 2016 Thirteenth International Conference on*, IEEE, 2016, pp. 1–5.
- [28] Z. Gui, Y. Wang, Y. Liu, and J. Chen, “Mobile visual recognition on smartphones,” *Journal of Sensors*, vol. 2013, 2013.
- [29] C.-S. Wang, S.-H. Hung, and D.-J. Chiang, “A markerless augmented reality mobile navigation system with multiple targets display function,” in *Applied System Innovation (ICASI), 2017 International Conference on*, IEEE, 2017, pp. 408–411.
- [30] L. Pombo and M. M. Marques, “Marker-based augmented reality application for mobile learning in an urban park: Steps to make it real under the edupark project,” in *Computers in Education (SIIE), 2017 International Symposium on*, IEEE, 2017, pp. 1–5.
- [31] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

- [32] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [33] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*, Springer, 2006, pp. 404–417.
- [34] G. Facciolo, N. Limare, and E. Meinhardt-Llopis, “Integral images for block matching,” *Image Processing On Line*, vol. 4, pp. 344–369, 2014.
- [35] C. Chen and C. Hsiao, “Haar wavelet method for solving lumped and distributed-parameter systems,” *IEE Proceedings-Control Theory and Applications*, vol. 144, no. 1, pp. 87–94, 1997.
- [36] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *European conference on computer vision*, Springer, 2006, pp. 430–443.
- [37] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*, Springer, 2010, pp. 778–792.
- [38] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Computer Vision (ICCV), 2011 IEEE international conference on*, IEEE, 2011, pp. 2564–2571.
- [39] C. Harris and M. Stephens, “A combined corner and edge detector.,” in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.
- [40] Statista, *Mobile os market share 2018*, May 2017. [Online]. Available: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>.
- [41] Google, *Arcore overview*. [Online]. Available: <https://developers.google.com/ar/discover/>.
- [42] Mapbox, *About*. [Online]. Available: <https://www.mapbox.com/about/>.
- [43] U. Technologies, *Unity3d*. [Online]. Available: <https://unity3d.com/unity>.
- [44] OpenCV, *About*. [Online]. Available: <https://opencv.org/about.html>.
- [45] Flask, *Welcome*. [Online]. Available: <http://flask.pocoo.org/>.
- [46] JSON, *Introducing json*. [Online]. Available: <https://www.json.org/>.

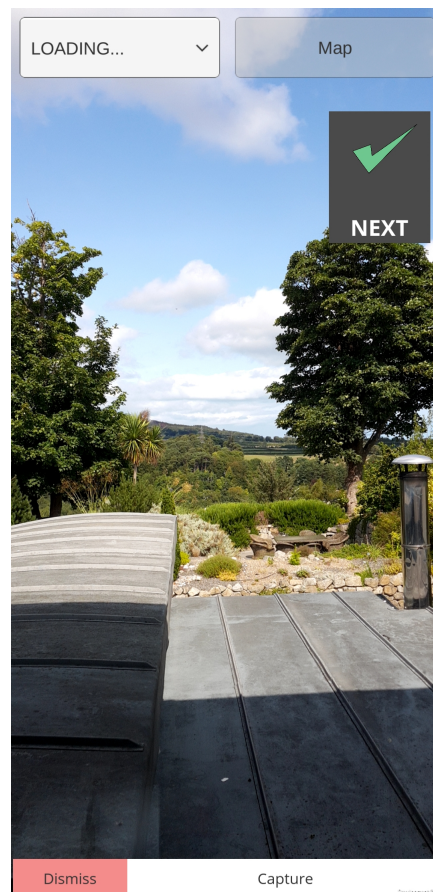
- [47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [48] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

# Appendix 1 - YouTube Video

A YouTube video demonstrating AR Campus Tour is available at:

[https://youtu.be/QM22uf\\_kUs8](https://youtu.be/QM22uf_kUs8)

# Appendix 2 - AR Campus Tour Walkthrough



*Figure 1: On start - the loading message in the Tour Select Menu indicates an ongoing API request*



*Figure 2: API request was successful - the Tour Select Menu asks users to select a tour*

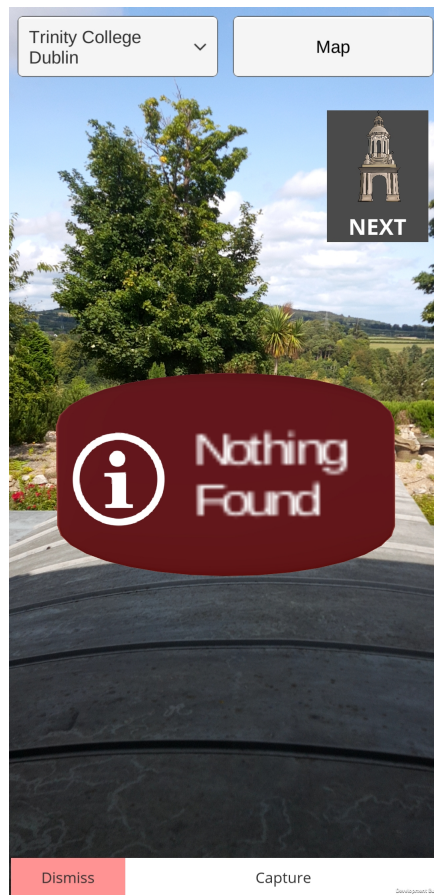




*Figure 3: Selecting a tour*



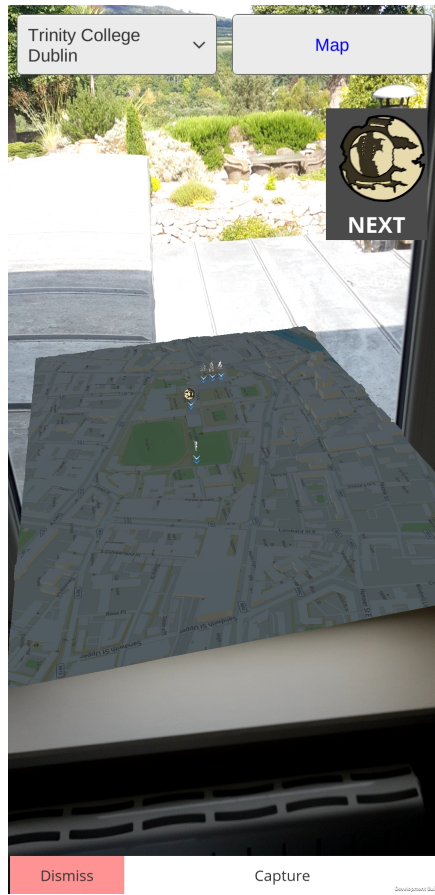
*Figure 4: A tour has been selected - the Map Button has become active and the Next Landmark Icon has been initialized*



*Figure 5: Search is unsuccessful - the Nothing Marker has appeared*



*Figure 6: Search is successful - the AR Landmark Marker has appeared and the Next Landmark Icon has been updated*



*Figure 7: Map is active*



*Figure 8: Map after a successful search - a green tick appears above the landmark position marker*

## Appendix 3 - Database Images



*Figure 9: Database Images of Landmark A - Sphere Within Sphere*



*Figure 10: Database Images of Landmark B - Walton Memorial Statue*





*Figure 11: Database Images of Landmark C - Statue of Lecky*



*Figure 12: Database Images of Landmark D - The Campanile*



*Figure 13: Database Images of Landmark E - Statue of Salmon*



*Figure 14: Query Images*