AUGMENTED REALITY IN RETAIL

Final Year Project
Abstract

Online shopping provides convenience, minimal setup costs and limitless inventory for retailers. While Physical stores provide an interactive and intuitive environment for customers to view products. This report explores the opportunity present in the two markets and proposes a hybrid shopping experience enabled by Augmented Reality technology and Photogrammetry. The vision would be manifested as an iOS based application that scans objects in 3D and allows convenient viewing in AR. The project aims to create a working e-commerce platform within the time frame of ten months that serves both customers and retailers.
Acknowledgement

I would like to thank our project supervisor Dr. HF Ting for his valuable feedback and support during the course of the project. I would like to thank the Computer Science Department for helping us procure iOS devices for testing. I would also like to thank professor Cezar Cazan for providing us with guidelines on writing a good project report.

Abbreviations

- AR: Augmented Reality
- SDKs: Software Development Kits
- UI: User Interface
- VIO: Visual Inertial Odometry
- IMU: Inertial Measurement Unit
- PGM: Photogrammetry
Table of Contents

List of Figures ................................................................. 4
List of Tables ................................................................. 4

1 Introduction ...................................................................................... 6
   1.1 Background ................................................................. 6
   1.2 Scope .............................................................................. 6
   1.3 Deliverables ............................................................... 6
   1.4 Outline ............................................................................. 6

2 Related Works .................................................................................. 7
   2.1 Ikea .............................................................................. 7
   2.2 Scann 3D ................................................................. 7

3 Features and Objectives .................................................................... 7
   3.1 Model Creation ........................................................ 7
   3.2 Product Demos ....................................................... 8
   3.3 User Interface .......................................................... 8

4 Methodology ...................................................................................... 8
   4.1 Technologies used ..................................................... 8
      4.1.1 Apple’s ARKit 2 ................................................... 8
      4.1.2 Unity 2018.1.1 ............................................... 9
      4.1.3 ReCAP API by Autodesk ...................................... 9
      4.1.4 Firebase Unity SDK .......................................... 9
   4.2 Application Design ....................................................... 9
   4.3 System Design ............................................................... 10
      4.3.1 Retailer part ....................................................... 10
      4.3.2 Buyer journey ................................................... 11

5 Current Status .................................................................................. 13
   5.1 Feasibility Study: Completed ....................................... 13
   5.2 ARKit Functionality Testing: Completed ...................... 13
   5.3 ReCAP API Functionality Testing: Completed ............... 14

6 Results and Discussion ................................................................... 14
   6.1 Surface Detection ...................................................... 14
   6.2 Ambient Light Detection ............................................. 14
   6.3 Rotation Angle ............................................................ 14

7 Next Steps ...................................................................................... 15

8 Conclusion ...................................................................................... 15

9 References ...................................................................................... 16
List of Figures
Figure 1: Retailer part...........................................10
Figure 2: Basic Object Spawning.........................12

List of Tables
Table 1 : Project Schedule………………………………15
1. **Introduction**

Augmented Reality today is a rather new technology with several implications in the modern world. However, as it is still very fresh, not much is known about its potential and implications. This section aims to implore several aspects of the technology with respect to the project at hand.

1.1. **Background**

The current retail landscape can be broadly divided into two categories-offline shops and online stores. Online stores have become a norm in the past few years due to minimal setup costs and no inventory costs. However the product viewing experience is not intuitive enough for buyers which often leads them to prefer Offline stores to make purchase decisions. Unfortunately for retailers, Offline stores still have infrastructure costs associated with them which makes them expensive to set up. Our project aims to bring the advantages of physical product viewing to online shopping using Augmented Reality and Photogrammetry. This hybrid shopping experience would enable users to interact with the products in virtually created environments that they can view from their phones.

1.2. **Scope**

The development of the retail application would be done on Apple’s ARKit2 platform thus the scope would be limited to iOS devices. The reason for choosing ARKit as our development platform has been discussed in the section 4.1.1. The application would be ready to deploy on the following ARKit Compatible devices.

- iPhone 6s and 6s Plus.
- iPhone 7 and 7 Plus.
- iPhone SE.
- iPad Pro (9.7, 10.5 or 12.9) – both first-gen and 2nd-gen.
- iPad (2017)
- iPhone 8 and 8 Plus.
- iPhone X, Xs and Xr

1.3. **Deliverables**

Our end deliverable would be an iOS based application that would be ready to deploy on the devices mentioned above. We would be using the Firebase SDK(refer to section 4.1.2) for database creation and the ReCap API for photo to model creation.

1.4. **Outline**

The report discusses the current trends in Augmented Reality and points out our motivation to engage in this project. This is followed by a list of objectives we aim to achieve and a detailed methodology for
executing the project. The report then explores the work we have done till now and the results obtained in the process. The next steps to the project are discussed along with a project schedule in the final section.

2. **Related Works**

AR has been used in mobile shopping applications by various firms like IKEA and Shopify to add competitive edge to their consumer-experience (Sheehan, 2018). Photogrammetry has been used in architecture, industrial design and game development industries. This technique of modelling uses photos to make measurements between objects and create a subsequent geometric representation of the same. Recently companies like Shopify have shown interest in this space by launching services for retailers to convert their products to 3D scans.

The following section discusses a few of these use-cases and also furthers our case for making an using the 2 technologies in our application.

2.1 **Ikea**

Ikea Place is an application that helps the user visualise the furniture in their home space before making a purchase decision. (Chang, 2018).” This is particularly important in the home decor retail space as customers often spend countless hours trying to decide whether something will look good in their homes or not. The “try before you buy” concept by Ikea has been very well received by its customers and serves as a motivation for us to build our application for at-home product viewing.

2.2 **Scann3D**

This application requires the user to take approximately 20 to 30 photos of the object that they wish to 3D scan. This application available on iOS makes the device a standalone tool to turn images into 3D models for storing, sharing and editing in Sketchfab.

3. **Features and Objectives**

This section outlines the different features of the application that we aim on building in the project.

3.1 **Model creation**

The retailer will be guided through a process to take photos of the object they want to sell. After the photosession is complete, our application would convert these photos to 3D models using PGM. These 3D objects can be used by the retailer to display the items he has for sale along with the pictures he/she took.
3.2 Product Demos
These items for sale inside the storefront would form the basis of our AR experience. The user would be able to interact with the objects in AR in a manner similar to physical products before making a purchase. Our project focuses on four types of products that can be viewed at homes - Jewellery, Furniture, Handicrafts and Artwork.

3.3 User Interface
To guide the user through different modes of the application, an easy to use and intuitive UI will be designed. While the overall layout of the application would be based on existing e-commerce platforms. Several unique design features will be introduced to emphasise the hybrid experience of the product.

4. Methodology
This section discusses the technologies used by the team to implement the project. It lays out the application design as well as the system design in detail.

4.1 Technologies used

4.1.1 ARKit Unity SDK

ARKit is a Visual Inertial Odometry (VIO) system that tracks the user’s position by matching a point in the real world to a pixel on the camera sensor at each frame refresh. The Inertial Measurement Unit (IMU) also measures the user’s position using inertial sensors - Gyroscope and accelerometer. The time measurements and accelerations are integrated backwards to calculate velocity and distance travelled between two frames. While error in such a system is upwards of 30% in most systems, ARKit’s error correction reduces the error to single digit. Every tiny muscle movement can be detected by the inertial system providing unparalleled accuracy. The output of these systems are combined using a Kaiman Filter to provide the best estimate of the actual position which is published to the ARKit SDK.

4.1.2 Unity 2018.1.1

Being a mature 3D game engine, Unity is the perfect choice for AR development. It has well documented functions and large developer community, making it easier to build applications. The Unity ARKit plugin wraps the SDK in C# scripts, this makes the code object oriented and allows for a modular software design. Unity also has tools like ARKit Remote that allow creating interactions directly in the Unity editor saving time on building the application each time.
4.1.3 ReCAP API by Autodesk
ReCap Photo is an Autodesk 360 service that creates 3D Scans from photos such that users can view and share the 3D data. Users upload images on Autodesk 360 and the Cloud-based service creates a 3D mesh model reading for importing. The API works on REST Calls and allows developers to integrate this service for into their applications and products.

4.1.4 Firebase Unity SDK
The application design requires dynamically created 3D files for rendering and display. Firebase is a great database platform that safely stores assets and user made content in the cloud. Its real-time database syncs content across all devices in a few hundred milliseconds. With its Unity SDK it becomes easy to query the database directly from the application itself and allows for the backend logic to be implemented in C# scripts.

4.2 Application Design
The current application consists of two users primarily, the shopkeepers and the customer. They would have access to a different set of classes and objects.

The shopkeeper would be able to setup his shop with details and add categories of items he/she sells. Inside each Product Category the user would add the items for sale with description, price, pictures and the model. If the user opts to create a model, he/she will be asked to either upload 20 pictures of the object or use the application to take pictures. These pictures would be converted to a 3D model which the retailer can add in the item display.

The Customer would be able to use the application in a similar manner to existing ecommerce applications. In the landing screen, the customer would be able to view the shopfronts and select a shop to visit. Inside the shopfront, the customer would view different product categories and select one to view the products on offer. When the user views a specific item, a button will popup in the screen to view the model of the object. On clicking, the user will be able to point the camera at a surface and the object will appear in 3D for interactive viewing.
4.3 System Design

4.3.1 Retailer Part

Figure 1

The Retailer enters their account with a username and password. They can create a shopfront object and add categories, and objects.

New shop entity is created in firebase database along with its various attributes.

In the TakePhoto() function the WebCamTexture API is used to take pictures and each image is saved as a 2D Texture. These textures are then converted to a byte array for uploading to firebase.

In case of the UploadPhoto() function, the .jpg photos are converted to a byte array and uploaded to firebase.

After all photos have been uploaded, database references(URLs) for the images are created and the following Calls are made to the ReCAP API for processing the mesh.
**Part of PhotoToMesh()**

- **POST photoscene**
  This creates a Photoscene that will perform calibration and meshing.

- **POST file**
  The reference for each image file is provided in this call for uploading to ReCAP360.

- **POST photoscene/:photosceneid**
  Initiates processing of the Photoscene created above.

**Part of GetModelStatus()**

- **GET photoscene/:photosceneid/progress**
  Returns the current processing progress percentage and status of the Photoscene.

- **GET photoscene/:photosceneid**
  Returns a time-limited HTTPS link to the .obj output file that can be downloaded for display as 3D object.

A Model class Object is created inside the GetModelStatus() function and the 3D object is assigned to a GameObject in the class. The GameObject is then converted to a byte array and uploaded to the database for easy retrieval and access in the other parts of the application.

**4.3.2 Buyer Journey**

Step 1: The user starts the application

Step 2: User browses through the list of shops which appear from the data entries

Step 3: User selects and enters the shopfront screen; selects a product from the list.

Step 4: A 3D Model is pulled from the database and the camera is started again.

Step 5: New navigation plane is displayed on which the selected object is placed.

Step 6: Repeat step 1,2,3,4,5 till user exits.

**4.4 Technical Implementation: Object Spawning**

When the user selects a product from the product list, the selected product is spawned as a game object on the AR Scene. ARKit’s WorldTrackingConfiguration provides the basis for this as it helps to map and track the user’s real world space. It allows you to create (i.e spawn) the game objects as a virtual content on the user’s rear camera view.
To achieve this, we used Unity’s ARKit plugin which is readily available on bitbucket as it is an open source content. The plugin is imported as an asset of the project. It has 5 example scenes. The one that we used for basic spawning is the UnityARKitScene. The scene is composed of various components which can be seen in the hierarchy section. The main components used for spawning are:

1. **Light type**: Light type is the component required for the desired lighting of the game object for it to be seen. There are three main light types:
   a. **Directional Light**: It refers to natural light or sunlight as it has a direction but doesn’t have a source location. It helps to create a shadow on the game objects. There are 3 shadow types available—no shadows, light shadows, and dark shadows. Other alternatives to light types include:
   b. **Ambient Light**: In this, the same amount of light hits the game object from all directions. This is used when the game object is not required to have any shadows.
   c. **Omni**: In this particular type, the light has a direction and a position.
   d. **Spot**: As the name suggests, this light type has a direction and a position but falls on the game object in a conical manner just like a spotlight.

2. **Game object component**: This will be the prefab asset that will be imported on user selection. It is the component which will be spawned.

3. **AR Camera Manager**: The Unity AR Camera Manager class allows for game object customisation. It is used for making adjustments to the following functions—Alignment, plane detection orientation, point cloud data, and light estimation. It links the device’s camera to this class so as to understand which camera is to be managed. It also uses swift scripts that are required to place the game objects on the user’s camera view.

4. **ARKit Control**

The build settings have to be switched to an iOS platform as ARKit isn’t compatible with any other unity platform. This scene is essential for object spawning. The components will be added/deleted according to the game object and will be rendered on the user’s real-world space. As an example, the figure below shows a “random cube” object spawned on the carpet as seen from the user’s rear camera.
Similarly, several asset prefabs can be placed in the hierarchy section. Transformation and scaling can be applied to the objects as required. The figure below involves a house as a game object to be spawned. Using the directional light component and the shadow plane (which is scaled according to the size of the prefab) the house has both internal and external shadows making it more life-like which a major requirement in an AR application.

How this spawning works on the backend is that when the plane detection as specified in the AR World Tracking Configuration is enabled, anchors are added and updated for each detected plane (the detected planes are represented by the blue rectangular frames in figure). The rendered method constantly updates itself when the device is moved providing a better and realistic augmented experience.

5. **Current Status**

This chapter will discuss and conclude the various milestones achieved by the team across the various development phases.
5.1 Feasibility Study: Completed
To understand the structure and components of the ARKit SDK, we performed a thorough analysis of the Apple Developer Documentation. We explored tutorials created by Unity and Udacity to understand the workflow for creating an AR application. The ReCAP API documentation was explored as well to understand how we would design a system architecture consistent with its usage.

It was essential for the team to be well-versed with C# before jumping into full scale development. Thus, the team spent considerable amount of time shifting from Java based development to C#. The feasibility study helped the team define the objectives and scope for the project more clearly.

5.2 ARKit Functionality Testing: Completed
The team performed preliminary plane detection tests in different environments to assess the platform’s reliability. These plane detection tests involve identifying feature points and loading prefabs into the plane function for rendering. After rendering, we tweaked texture values and created shaders to make the plane appear better.

After plane detection tests were completed, the team tried to spawn objects using Raycasting functions. These objects were sample 3D models downloaded from https://free3d.com/ and stored as assets in Unity. The objects were placed in a Gameobject list and were spawned randomly to check the accuracy of the Raycasting.

5.3 ReCAP API Functionality Testing: Completed
The first step involved creating the functions to open the device camera and save the pictures. This was done by creating a RawImage UI object in Unity that would display the camera’s view. Unity captured a 2D texture in the object which had to be further converted to bytestream using Encode2PNG function. These bytestreams were saved and this array was then sent in a REST POST Call to the ReCAP360 for model creation with the photos. Subsequent GET Calls were made to check the status of the mesh processing and after 15 minutes the files were ready and downloaded onto the system. We viewed the objects on Blender to check the mesh quality and troubleshoot ways of improving it.

6. Results and Discussion

6.1 Surface detection
The testing for plane/tabletop detection gave almost perfect results on smooth surfaces but failed on surfaces with an uneven texture. Although ARKit2 has the most advanced VIO detection/tracking its performance is limited to certain types of environments and cannot perform well in rugged environments. This issue isn’t a mere software engineering problem but one that depends on the device’s camera quality and specifications. This led us to add an introductory guide in the beginning of the application which advises the user to select smooth surfaces for plane detection in order for the application to run effectively.
6.2 Ambient light detection
During object/model testing, sometimes the objects appeared dull with very low brightness. To fix this issue, we created a new function (as defined in the code snippet below) which auto adjusts the lighting of the object by making use of the room’s ambient lighting. Figure 9 shows the difference between ordinary image rendering and image rendering with Ambient Light Detection.

```c
void fixlighting {
    float new_light = camera.lightData.arlightEstimate.ambientIntensity / 1000;
    return new_light;
}
```

6.3 Rotation angle
With Unity being a game engine, the device camera API was poorly documented and we encountered problems in rotating the image as the device rotated. To solve this problem, we added a function to check the rotation angle in each updated frame and scale the RawImage background accordingly.

```c
float scaleY = backCam.videoVerticallyMirrored ? -1f : 1f;
background.rectTransform.localScale = new Vector3(1f, scaleY, 1f);
int orient = -backCam.videoRotationAngle;
background.rectTransform.localEulerAngles = new Vector3(0, 0, orient);
```

7. Next Steps
Having completed the feasibility study and the functionality testing, the team is currently building the software to integrate the creation and display parts.

<table>
<thead>
<tr>
<th>Schedule</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>September-October 2018</td>
<td>Feasibility Study</td>
<td>Completed</td>
</tr>
<tr>
<td>November-December 2018</td>
<td>Functionality Testing</td>
<td>Completed</td>
</tr>
<tr>
<td>January-February 2019</td>
<td>System Design and Backend Development</td>
<td>In Progress</td>
</tr>
<tr>
<td>March-April 2019</td>
<td>UI and Testing</td>
<td>Planned</td>
</tr>
</tbody>
</table>

[Table 1]

8. Conclusion
Augmented Reality is still in its infancy and has a lot of potential to grow in the future. With companies like Apple and Google developing dedicated software bundles for AR, developers are creating world-class AR applications. Our aim to bridge the gap between physical and online stores is a humble attempt at building a strong use case for AR for mobile devices. Though our shops are limited in the number of items they sell, we only aim to create a proof-of-concept that can serve as a guide for future development. We hope to implement the payment portals and other features once the project objectives are achieved and a product is created. Our project isn’t just a one-off application but a new kind of user experience that uses visually imposed images as interaction points. We believe that the application has the potential of changing the retail industry and how people perceive AR applications.

References
