

SAMRIDDHI COLLEGE
Tribhuvan University
Institute of Science and Technology



**A PROJECT REPORT
ON
AR BASED TRY ON TECH**

Submitted To:
Department of Computer Science and Information Technology
Samriddhi College

*In partial fulfillment of the requirement for the Bachelor Degree in Computer Science
and Information Technology*

Submitted By:
Sandesh Pokharel(16172/074)
Jeevan Karki(16148/074)
Simanta Poudel(16176/074)

Under the supervision of
Mr. Loknath Regmi

April 29, 2022

SAMRIDDHI COLLEGE
Tribhuvan University
Institute of Science and Technology



**A PROJECT REPORT
ON
AR BASED TRY ON TECH**

Submitted To:
Department of Computer Science and Information Technology
Samriddhi College

*In partial fulfillment of the requirement for the Bachelor Degree in Computer Science
and Information Technology*

Submitted By:
Sandesh Pokharel(16172/074)
Jeevan Karki(16148/074)
Simanta Poudel(16176/074)

April 29, 2022



Samriddhi College
[T.U. Affiliated]
Lokanthali-16, Bhaktapur

Supervisor Recommendation

I hereby recommend that this project entitled "AR Based Try On Tech" is prepared under my supervision by Sandesh Pokharel, Jeevan Karki and Simanta Poudel in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology is processed for the evaluation.

.....
Mr. Loknath Regmi
Supervisor
CSIT Department
Samriddhi College
Lokanthali, Bhaktapur
Nepal



Letter of Approval

I hereby recommend that this project is prepared under my supervision by Sandesh Pokharel, Jeevan Karki and Simanta Poudel entitled "AR Based Try on Tech" in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

.....
Mr. Loknath Regmi
Supervisor
CSIT Department
Samriddhi College
Lokanthali, Bhaktapur
Nepal

.....
Mr. Sandeep Shrestha
Principal
Samriddhi College
Lokanthali, Bhaktapur
Nepal

.....
Signature of External Examiner
Institute of Science and Technology (IoST)
Tribhuvan University
Nepal

Acknowledgement

We would like to express our sincere gratitude to every individual including teachers, family and friends who has helped us in our project with their feedback and suggestions.

We would like to give special thanks to **Mr. Sandeep Shrestha**, Principal for providing us opportunity to undertake this project as part of the curriculum.

We would like to acknowledge with much greater appreciation for the crucial role of our supervisor **Mr. Loknath Regmi**. Without his guidance and support, this project wouldn't have been possible. His constant encouragement and help is what mostly helped us to complete this project.

Lastly, we would like to appreciate the hardwork and commitment of every team members including **Sandesh Pokharel, Jeevan Karki, Simanta Poudel**. The effective co-ordination and dedication of each member helped in the successful completion of the project.

Thank You,
Sandesh Pokharel(16172/074)
Jeevan Karki(16148/074)
Simanta Poudel(16176/074)

Abstract

One of the biggest challenges of online shopping is that it doesn't help much for a complete sensory product experience. In the real and physical environment, you can try on clothes, touch the fabric, and see for yourself how the apparels looks on you. While these things are technically impossible in e-commerce, augmented reality (AR) applications provide customers with a way to provide customers with deeper and more comprehensive information about their products from home. So, we have used Vuforia which uses underlying feature matching algorithm SIFT (Scale Invariant Feature Transform) to construct the image target and then match the extracted features to the individual frames of real time video feed and then if matched, augment the 3D shoe model. It helps to make it AR based try-on possible.

Keywords: *Augmented Reality, AR, AR Shopping, SIFT, Feature matching, Pattern Matching, Gaussian Blur, Vuforia, Unity, E-commerce*

Table of Contents

Acknowledgement	v
Abstract	vi
Table of Contents	vii
List of Figures	ix
List of Tables	x
List of Abbreviations	xi
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	1
1.3 Objectives	2
1.4 Scope and Limitations	2
1.5 Report Organization	3
Chapter 2: Literature Review	4
Chapter 3: System Analysis	7
3.1 Requirement Analysis	7
3.1.1 Functional Requirements	7
3.1.2 Non-Functional Requirements	8
3.2 Feasibility Study	9
3.2.1 Technical Feasibility	9
3.2.2 Operational Feasibility	9
3.2.3 Economic Feasibility	10
3.2.4 Schedule Feasibility	10
Chapter 4: System Design	11
4.1 Design	11
4.1.1 Architecture Design	11

4.1.2	Sequence Diagram	12
4.2	Algorithm Details	13
Chapter 5: Implementation and Testing		17
5.1	Implementation	17
5.1.1	Tools Used	17
5.1.2	Implementation Details Of Modules	17
5.2	Testing	27
5.2.1	Unit Testing	27
5.2.2	Integration Testing	31
5.2.3	System Testing	32
5.3	Result Analysis	35
Chapter 6: Conclusion and Future Recommendations		36
6.1	Conclusion	36
6.2	Future Recommendations	36
References		37
A. Appendix		38

List of Figures

Figure 3.1.1	Use Case Diagram	8
Figure 3.2.1	PERT Chart	10
Figure 4.1.1	Architecture Design	11
Figure 4.1.2	Sequence Diagram	12
Figure 4.2.1	SIFT Algorithm	13
Figure 8.0.1	Vuforia model target data for unity	38
Figure 8.0.2	Creating guided view for target model	38
Figure 8.0.3	Creating AR Camera for unity using vuforia package	38
Figure 8.0.4	Replacement model	38
Figure 8.0.5	Placing replacement model on top of target model	39
Figure 8.0.6	Try it out button	39
Figure 8.0.7	Trying out the shoe app	40

List of Tables

Table 5.2.1	Unit Testing - Case 1	28
Table 5.2.2	Unit Testing - Case 1	28
Table 5.2.3	Unit Testing - Case 2	29
Table 5.2.4	Unit Testing - Case 2	30
Table 5.2.5	Integration Testing - Case 1	31
Table 5.2.6	Integration Testing - Case 2	31
Table 5.2.7	Test Case: Without Wearing Shoes	32
Table 5.2.8	Test Case: With Wearing Shoes	33
Table 5.2.9	Test Case: With Another Shoe	34

List of Abbreviations

3D	3-Dimension
AR	Augmented Reality
GPU	Graphics Processing Unit
LiDAR	Light Detection and Ranging
MOIAR	Multi Object Identification Augmented Reality
MTG	Model Target Generator
Open CV	Open Source Computer Vision
RLO	Reusable Learning Objects
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transform
UML	Unified Modeling Language

Chapter 1: Introduction

1.1 Introduction

Augmented Reality is real-time direct or indirect picture of a physical real-world environment that has been enhanced/augmented by the addition of virtual computer-generated information.

AR is a developing trend among businesses that deal with mobile computing and commercial apps. AR enhances the user's perception of and interaction with the real world. Augmented reality uses advanced technology like object recognition and computer vision to make information from the actual world interactive for the user. Researches are being carried out in a massive scale to apply Augmented reality in different commercial sectors.

Transforming the customer experience fully with augmented reality. Customers can try on glasses, clothes, shoes, and watches on your website. Augmented reality technology provides an immersive experience with 3D visualization, virtual fitting, and product demos, so AR gives consumers a unique opportunity to try them out before buying. In this way, augmented reality technology allows customers to make informed purchases, helping to counteract high rates of return and ultimately leading to lower rates of return.

With AR based try-on tech, customers can discover new releases as well as classic sneakers. They can try the sneakers on their feet right away, no matter where they are, and can then take a photo of them wearing some branded sneakers to surprise their friends.

1.2 Problem Statement

E-commerce sites have greatly enhanced conveniences of customers by helping them with buying products from the comfort of their own homes. Even then the rate of e-commerce product returns rose by 70% in 2020, as per Narvar. [1] Customers cannot try the clothing items online and therefore wouldn't know if it will suit them or not when they wear it physically. So, most often when they try out the products physically,

it won't suit them and eventually they will return the product. And the product return is a major headache for the retailers. So, with our AR based try-on tech, customers will be able to wear those items virtually in a real time environment using which they can decide if those clothing items is a right suit for them or not. Since customers can try out products beforehand with the option to choose different product colors, it is less likely that they will return the product once ordered.

1.3 Objectives

To help online shoppers understand what they are buying and if the product suits them or not by allowing them to virtually try it on.

1.4 Scope and Limitations

Augmented Reality is quite a trendy topic these days, especially with the announcement of Meta(formerly Facebook) to go all out to build the Metaverse. The opportunities are immense in the field of AR. AR has huge potential in the fields of gaming, entertainment, health, education, tourism, and the fields will increment more in the coming days. AR based try on tech in e-commerce has tremendous opportunities given that the e-commerce customers are forever wanting a product that could show them beforehand if the apparels properly fit and suit them or not before buying the product. However, there are certain limitations as listed below:

- System might not provide accurate results.
- System might not still help users to determine the quality as apparels cannot be touched.
- System might not still help users to determine the comfortability of the product.

1.5 Report Organization

The report on project "AR based try on tech" is categorized into six different chapters.

Chapter 1: This chapter describes the introduction, objectives, scope and limitations of the project.

Chapter 2: It is the review of the past research works of the system.

Chapter 3: It explains the functional and non-functional requirements of the system and also includes the feasibility study.

Chapter 4: It consists of the UML diagrams and algorithm used in the project.

Chapter 5: It includes the implementation details and tools and technologies used to develop the project. It also includes testing of the system.

Chapter 6: This last chapter includes the conclusion and future recommendations for future enhancements.

Chapter 2: Literature Review

Unity3D and Vuforia can be used to create an augmented reality application that does not require the use of markers. Vuforia is a mobile augmented reality software development kit (SDK) that allows AR applications to recognize objects, track the surroundings in real time, and apply digital interactive augmentations. A survey of the literature on the subject revealed that several similar techniques had been used, ranging from the detection of feet to the placing of virtual items in the real environment. Based on the research, a framework for the complete system's flow has been built, from foot tracking to enhancing items.

In the research, user studies found that AR-based try-on outperformed VR-based try-on, particularly in the 3D representation of apparel. Using AR technology, the fitness effect is made more realistic by the real-time interactive environment. AR-based try-ons with customised avatars can improve a product's 3D visualization. According to the conclusions of the study, AR-based try-on has more promise in the future of online purchasing. [2]

The authors of the research [3] have developed a very reliable method for detecting feet, which can be used as a foundation for future footwear augmented reality projects. To recognize a person's foot in an input image, the approach developed in this thesis uses image enhancement, background reduction, and an active contour model with specific types of images. The algorithm is fed an image of a person's foot taken in a controlled lighting situation. The image is enhanced, the foreground is extracted, and the active contour model is used to generate the outline of the foot in the image. Although the method developed has few hardware requirements, it does have several environmental constraints, such as:

- The lighting around the image should leave minimal visible shadows.
- The background images are to be close to identical as possible where the foot is to be detected.

The paper's authors [4] describe a deformable template matching with dynamic pro-

gramming for smart phone foot tracking. Instead of using an edge picture that only comprises the foot's border points, the researchers developed an algorithm that extracts the foot feature from a gray scale image template. The deformable template is made up of a series of segments that are connected and organized. The approach allows for greater flexibility in the deformation of particular items that demand it. The best match with the picture pixels determines the deformation of the template segments. The technique works well at various angles that are picked at random. The algorithm works well in cloudy days, interior lighting, and even dim light, however it fails in really dim or dark light. [5] The authors of this study developed a location-based object identification algorithm that was used to identify learning objects in a 5R adaptive location-based mobile learning environment. The 5R adaptation idea for location-based mobile learning states that the right material is delivered to the right learner at the right time, in the right location, through the right device (Tan, et al, 2011). By matching the tagged location information of the RLOs with the current location and orientation of the mobile device, the system intends to detect real-life learning objects. Furthermore, the algorithm has the capacity to guide the learner to the appropriate RLO for learning among the neighboring RLOs. The capacity to combine digital and physical material opens up the possibility of augmented reality learning, which generates excellent conditions for locative, contextual, and situation-based learning scenarios. Furthermore, we now have more tools and technology at our disposal to deploy mixed reality learning scenarios that deliver rich and immersive AR information, possibly reshaping how people and groups approach learning and education. Although the MOIAR software can identify many learning items at once, the screen space on a mobile device is restricted.

The authors of the research [6] have proposed ways to display larger-than-screen things on mobile devices by incorporating virtual elements such as messages, objects, or sensory stimulation into daily life. During the initialization process, the system uses a marker. It establishes a reference frame and acquires the camera's location and orientation in this way. By detecting points that appear in at least two frames, the algorithm recognizes the relationship between the video frames. The system reconstructs the 3D information from these points and uses it to create a map of the surroundings. The marker less tracking takes over when the marker is no longer visible. Using the points in the map that are present in the frame it gathers, the system guesses the camera's posi-

tion and orientation in the frame. It updates the map in order to collect new data points for future predictions. It, on the other hand, only functions in unprepared situations and relies on a camera as a sensor. The resolution of the camera is comparable to that of a mobile phone. It's possible that the system won't be able to connect a frame to the ones before it, causing the tracking to fail. The system isn't as quick or exact as it could be.

The authors of this study [7] describe AR manipulation techniques that use a mobile phone with AR capabilities as an interaction device. The paper demonstrates a number of virtual object translation and rotation algorithms. For virtual rotation, the authors created a tactile interface paradigm. When using the physical input method, the virtual model is fixed in relation to the phone and can be positioned or translated; additionally, when the model is de-selected, it returns to its original location. Objects are positioned, rotated, or translated for each fraction of a second while the buttons are pressed in the keypad approach. This strategy allows the user to rotate and translate more quickly. OpenGL was used to implement all of these techniques. The keypad manipulation produced precise and quick results. There were concerns with accuracy during the rotation, and more precise inputs were required to align the models accurately.

The different papers reviewed showed different perspectives and ways to build the system. However, template matching algorithm SIFT[4] can be used to build out the system as it is fast, accurate and precise for matching and also due to it being the underlying algorithm of Vuforia which will be used to build out the system.

Chapter 3: System Analysis

3.1 Requirement Analysis

3.1.1 Functional Requirements

Functional requirements describe the behavior of the overall functionality of the system. It specifies what a piece of software must do and how it must react to user input. Functional requirements of the system are listed below:

- Allow users to register and login to the system.
- Show a wide array of products catalog
- Allow users to view a product and click a button 'AR try on'
- Show 3D model of a product augmented to their foot

The use case diagram of the system is given below:

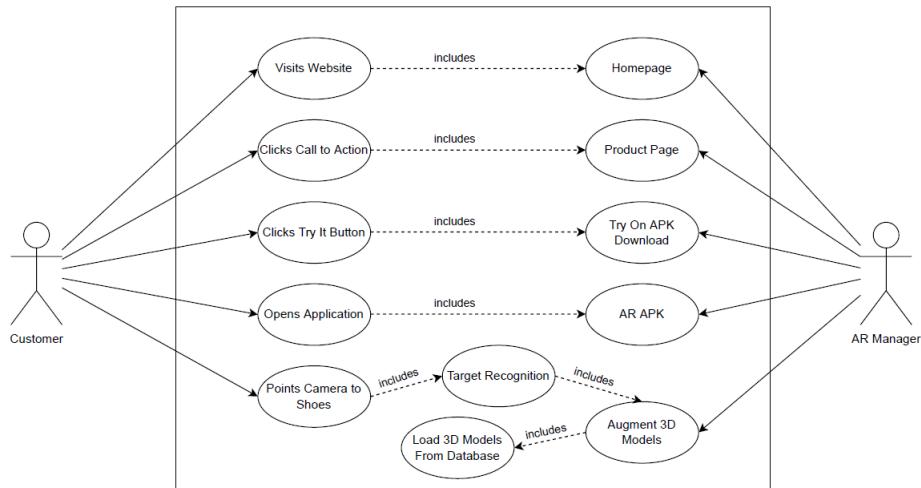


Figure 3.1.1: Use Case Diagram

3.1.2 Non-Functional Requirements

A Non-Functional Requirement describes a software system's quality attribute. The non functional requirements of our system are as follows:

Performance

Many systems require a high level of speed. As long as the hardware is in good condition, the application will process the output in no more than 30 seconds.

Availability

For the system to work, the server must be available 24 hours a day, 365 days a year.

Reliability

The system must have a high level of dependability. The server connection must be reliable and accomplish the intended activity. Processing may take 1-2 minutes longer if there is a lot of traffic.

Usability

The system is simple to comprehend, operate, add requirements, and interpret the output created because the user interface is pretty straightforward. As a result, the user can run

the system in a matter of minutes.

Maintainability

The system can be efficiently maintained to improve it as per the change in requirements.

3.2 Feasibility Study

A feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. After feasibility study was carried out, it was found that the system was feasible. The following types of feasibility studies were performed:

3.2.1 Technical Feasibility

It includes the technical requirements utilized to build the system.

- Software Requirements: Python and different libraries such as numpy, Open CV are utilized in the project that are simple to use and faster than most other languages.
- Hardware Requirements: The project requires GPU enabled processor machine with at least 4GB of memory and 15GB of secondary memory. AR apk requires Android 6.0 and above.

3.2.2 Operational Feasibility

The user is presented with UI of homepage which includes try it out button on each product section, which when clicked, an AR apk is downloaded which opens up the camera and augments 3D model to the foot. The system is simple to operate, thus making it operationally feasible.

3.2.3 Economic Feasibility

The system to be constructed in our project is a web application that requires minimum hardware and software support, much like any other ordinary web application. Additionally, some maintenance expenditures may be included as system integration fees. Aside from that, all of the other tools and technologies that will be employed in this system are open-source, making it cost-effective.

3.2.4 Schedule Feasibility

The project was broken down into smaller activities that were well distributed among the team members with proper scheduling ensuring the project completion on time.

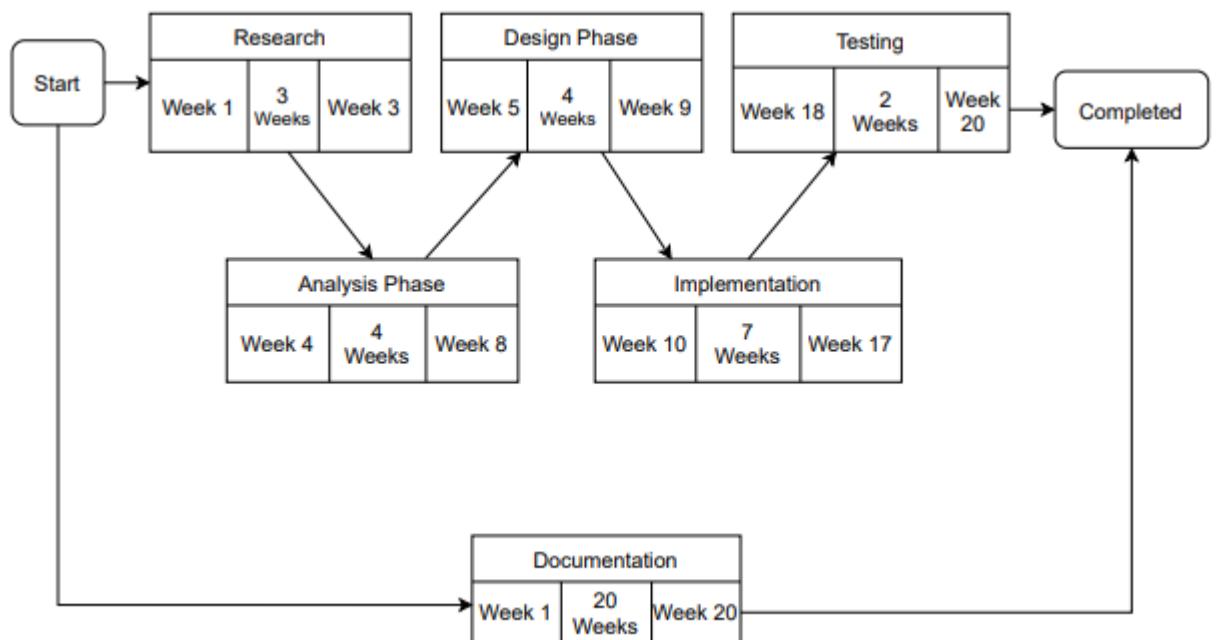


Figure 3.2.1: PERT Chart

Chapter 4: System Design

4.1 Design

The system allows the users to visit the UI homepage and click the try it out button to download the AR apk to augment the 3D shoe model to their foot.

4.1.1 Architecture Design

Architecture Design includes the architecture of the whole system which describes how the system is built from start to finish.

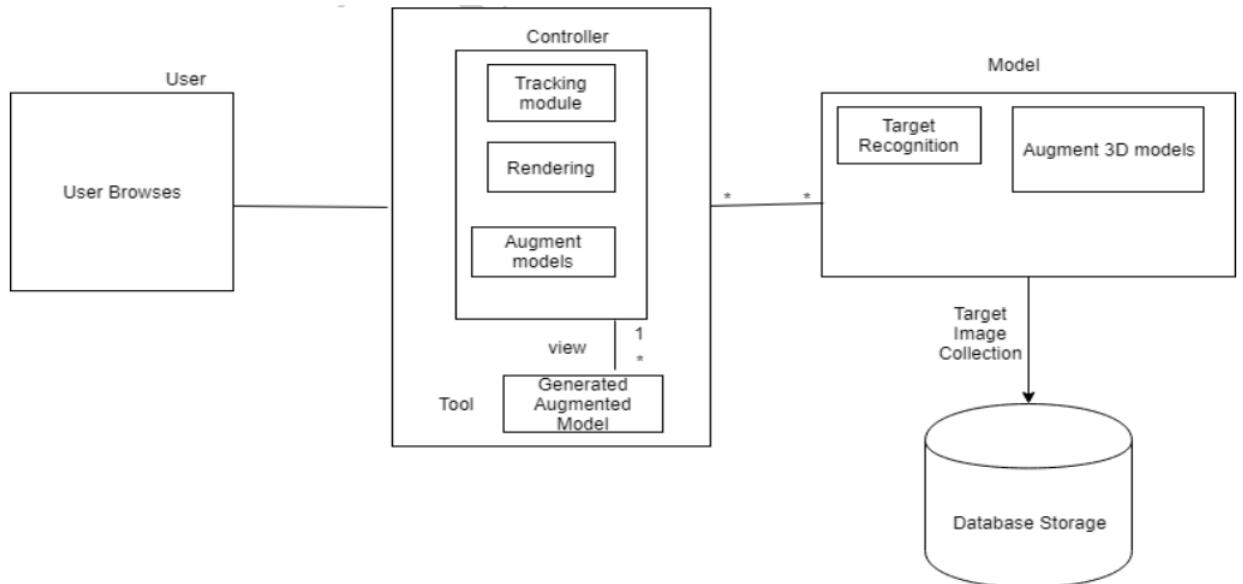


Figure 4.1.1: Architecture Design

4.1.2 Sequence Diagram

A sequence diagram shows the interactions between the objects of the system in a sequence.

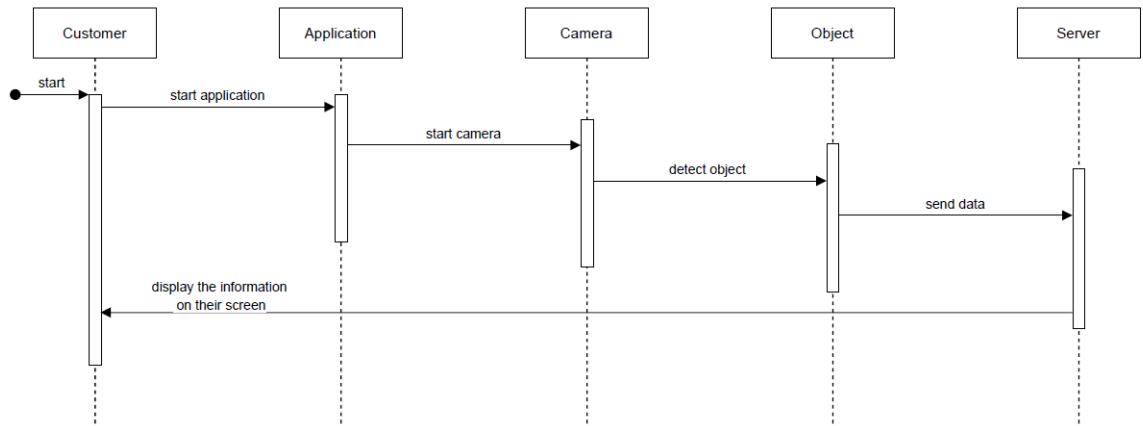


Figure 4.1.2: Sequence Diagram

4.2 Algorithm Details

SIFT Algorithm

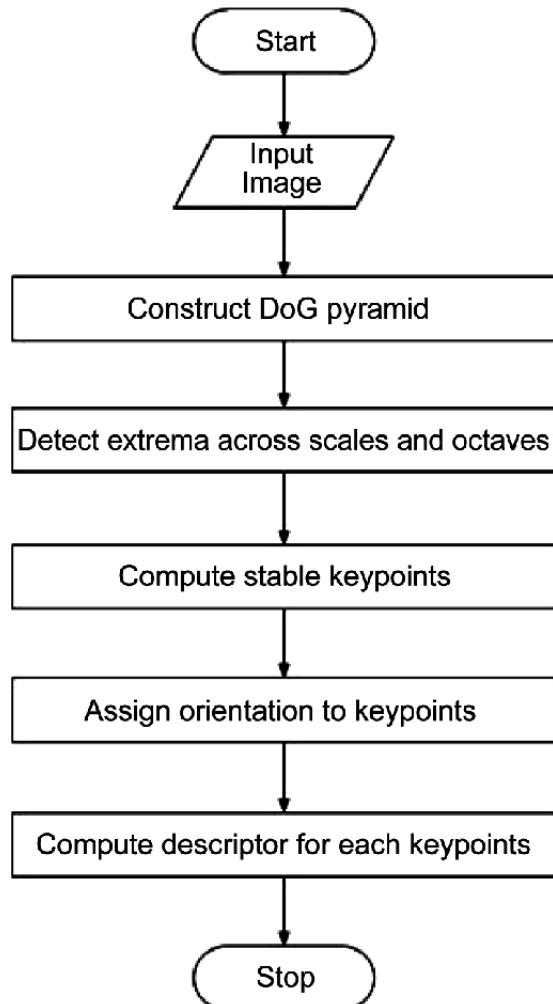


Figure 4.2.1: SIFT Algorithm

1. Create a image target by extracting features (keypoints) from the image using feature matching algorithm which is SIFT in our case.

SIFT:

- **Construct A Scale Space To Make Sure That Features Are Scale-Independent.**

Points of interest are detected which are termed keypoints in the SIFT framework. The image is convolved with Gaussian filters at different scales to reduce the noise in the image and to make the images scale-invariant, and

then the difference of successive Gaussian-blurred images are taken which is known as Difference of Gaussian(DoG) which is a feature enhancement algorithm that enhances the features of the images. Specifically, a DoG image $D(x, y, \sigma)$ is given by:

where $L(x, y, k\sigma)$ is the convolution of the original image $I(x, y)$ with the Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$, i.e.

- Keypoint Localisation For Identifying The Suitable Features Or Key-points.

Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images across scales. This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint. Since, many candidate keypoints are generated, many of them may be unstable, meaning points which are low contrast (and are therefore sensitive to noise) or poorly localized along an edge.

- **Discarding Low-Contrast Keypoints**

To deal with the low contrast keypoints, a second-order Taylor expansion $D(x)$ is computed for each keypoint. If the resulting value is less than 0.03 (in magnitude), we reject the keypoint. This Taylor expansion is given by:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x} \dots \dots \dots \quad (2)$$

where D and its derivatives are evaluated at the candidate keypoint and

$$\mathbf{x} = (\mathbf{x}, y, \sigma)^T \dots \dots \dots \dots \dots \dots \dots \quad (3)$$

- **Eliminating Edge Responses**

Remaining keypoints are checked to identify if they are poorly located.

These are the keypoints that are close to the edge and have a high edge response but may not be robust to a small amount of noise. A second-order Hessian matrix is used to identify such keypoints which is given as:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \dots \quad (4)$$

- **Orientation Assignment To Ensure The Keypoints Are Rotation Invariant**

An orientation is assigned to each of these stable keypoints so that they are invariant to rotation. At first, magnitude and orientation for every pixel value is calculated. For this, gradients in x and y directions are calculated which is given as:

$$Gx = (x + 1, y) - (x - 1, y) \dots \quad (5)$$

$$Gy = (x, y - 1) - (x, y + 1) \dots \quad (6)$$

Then, magnitude and orientation are calculated using the following formulae:

$$\text{Magnitude} = \sqrt[(G_x)^2 + (G_y)^2] \dots \quad (7)$$

$$\Phi = \text{atan}(Gy/Gx) \dots \quad (8)$$

An orientation histogram with 36 bins is formed, with each bin covering 10 degrees. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. In the case of multiple orientations being assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.

- **Keypoint Descriptor To Assign A Unique Fingerprint To Each Key-point.**

The neighboring pixels, their orientations, and magnitude, are used to generate a unique fingerprint for each keypoint called a ‘descriptor’. Since the surrounding pixels are used, the descriptors will be partially invariant to illumination or brightness of the images. A 16×16 neighborhood around the keypoint is taken. This 16×16 block is further divided into 4×4 sub-blocks and for each of these sub-blocks, the histogram is generated using magnitude and orientation. At this stage, the bin size is increased and only 8 bins are taken (not 36). Each of the arrows represents the 8 bins and the length of the arrows define the magnitude. So, a total of 128 bin values are generated for every keypoint. Although the dimension of the descriptor, i.e. 128, seems high, descriptors with lower dimension than this don’t perform as well across the range of matching tasks.

These keypoints are then used for feature matching.

2. Store the image targets along with its features in the database
3. From the video feed of the user camera, individual frames’ keypoints are extracted and matched with the image targets of the database using Sift algorithm.
4. Using those matched keypoints, augment the 3D shoe model to those matched keypoints.

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

Various tools are used to convert the requirements and design into implementation such as:

- Programming Language: Python is used to implement our proposed algorithm SIFT.
- Polycam is an ios lidar application that is used to scan 3D objects and extract a 3D model out of those scanned objects.
- Vuforia is a AR kit for mobile devices that helps to create AR applications. Vuforia includes developer portal for model target generation which helps to create model target out of 3D model generated by Polycam app.
- Unity is a game engine that uses vuforia package for integrating vuforia model target database and placement of 3D objects.

5.1.2 Implementation Details Of Modules

SIFT Implementatioin Using Python:

For implementation of SIFT (Scale Invariant Feature Transform), several things need to be calculated such as keypoints and description of the target image as well as the keypoints and description of the test image to match the features and get results based on them. To make this process simple and optimized, the different process and steps involved are broken into different module which is computed by our main module and function called

computeKeypointsAndDescriptors. Through this main module, it is easy to compute

the keypoints and descriptors to find the unique and absolute feature of any target or test images.

Sigma and k*sigma value as proposed in the algorithm are used to apply the blur using **Gaussian blur** on the gaussian kernel which is used by while generating the blur images.

```
float_tolerance = 1e-7

def computeKeypointsAndDescriptors(image, sigma=1.6, num_intervals=3,
assumed_blur=0.5, image_border_width=5):
    image = image.astype('float32')
    base_image = generateBaseImage(image, sigma, assumed_blur)
    num_octaves = computeNumber0fOctaves(base_image.shape)
    gaussian_kernels = generateGaussianKernels(sigma, num_intervals)
    gaussian_images = generateGaussianImages(base_image, num_octaves,
                                                gaussian_kernels)
    dog_images = generateDoGImages(gaussian_images)
    keypoints = findScaleSpaceExtrema(gaussian_images,
                                       dog_images, num_intervals, sigma, image_border_width)
    keypoints = removeDuplicateKeypoints(keypoints)
    keypoints = convertKeypointsToInputImageSize(keypoints)
    descriptors = generateDescriptors(keypoints, gaussian_images)
    return keypoints, descriptors
```

The first Step after the image is converted into float32 is to generate base image from the target image. This is an important function before getting the gaussian Images for furture process, which helps every image to be smooth and sizable.

In this process the image is resized by using the **cv2** package from Open CV where we pass in the image along with our center point i.e. from the origin (0,0) , fx (scale factor along the horizontal axis), fy(scale factor along the vertical axis), a flag that accepts one of the techniques listed below:

- INTER NEAREST is an interpolation based on the nearest neighbor.

- INTER LINEAR is a two-dimensional bilinear interpolation (used by default)
- INTER AREA - pixel area relation resampling. Because it produces moire-free outputs, it may be a preferable method for picture reduction. When the image is zoomed, however, the method behaves similarly to the INTER NEAREST method.
- INTER CUBIC is a bicubic interpolation over a 44-pixel area.
- INTER LANCZOS4 is a Lanczos interpolation over an 8x8 pixel area.

Here the two-dimensional bilinear interpolation is used i.e. INTER LINEAR to apply the linear interpolation first in one direction, and then again in the other direction. Using the numpy, the sigma difference value is found to be applied for the **Gaussian Blur**. The formula for calculating the sigma difference from our assumed blur level and k*sigma value is given as : $\text{sigma}_{\text{difference}} = \sqrt{\max((\text{sigma}^*2) - ((2*\text{assumed}_b\text{lur})^*2), 0.01)}$ This difference in value from sigma and assumed blur level is then used to apply the gaussian blur to the image using cv2. GaussianBlur function along the origin (0,0) on both the axis(x, y).Gaussian Blur removes the noise from the images to highlight the important features of the image.

```
from cv2 import resize, GaussianBlur, INTER_LINEAR
from numpy import sqrt
def generateBaseImage(image, sigma, assumed.blur):
    logger.debug('Generating base image...')
    image = resize(image, (0, 0), fx=2, fy=2,
                  interpolation=INTER_LINEAR)
    sigma_diff = sqrt(max((sigma ** 2) - ((2 * assumed.blur)** 2), 0.01))
    return GaussianBlur(image, (0, 0), sigmaX=sigma_diff,
```

For obtaining the DoG(Difference of Gaussian) images firstly the number of octaves are obtained i.e. how many times do it is needed to scale the image and how many subsequent blur images need to be created for each scaled image which is usually taken

as 4 times but the exact number of octave in our case is determined by taking the image shape from our base image as given below:

```
from numpy import round, log
def computeNumber0fOctaves(image_shape):
    return int(round(log(min(image_shape)) / log(2) - 1))
```

There are number of interval from our main module i.e **computeKeypointsAndDescriptors**. The number of images per octave for Image scaling is given by number interval given plus 3. Assuming default value to be 3 the number of images per octave would be 6.

With that it is needed to find the k*sigma value for figuring out the gaussian kernel. Gaussian Kernels for all the number of images per octave is firstly calculated through the zeros from cv2 which gives the scale of gaussian blur necessary to go from one blur scale to another in a single octave.

The first gaussian kernel is firstly declared by the initial sigma value within the module. A loop is then introduced from 1 to the number of images per octave i.e. also to gaussian kernels. For each item in the loop, the gaussian kernel is determined for the image by the square root of the total sigma value exponent of 2 difference the sigma of previous index exponent of 2. The list of all the gaussian kernels are figured from this process.

```
def generateGaussianKernels(sigma, num_intervals):
    logger.debug('Generating scales...')
    num_images_per_octave = num_intervals + 3
    k = 2 ** (1. / num_intervals)
    gaussian_kernels = zeros(num_images_per_octave)
    gaussian_kernels[0] = sigma
    for image_index in range(1, num_images_per_octave):
        sigma_previous = (k ** (image_index - 1)) * sigma
        sigma_total = k * sigma_previous
        gaussian_kernels[image_index] = sqrt(sigma_total ** 2
                                              - sigma_previous ** 2)
    return gaussian_kernels
```

Now that there are number of octaves and the gaussian kernels for each image for an octave. So, the next step is to figure out all the scale-space pyramid of the gaussian image. For this a loop is introduced 1 to the number of octaves obtained. The first image of each octave already has the correct blur applied so the first item of the gaussian images will be the same, but for the other images we need the value of each gaussian kernel respectively. Image is taken by applying GaussianBlur from cv2 to get the blurred image by applying the kernel for both dimensions(X and Y) respectively.

A octave base is an image which is determined by the 3rd last image after the loop is finished, as there is number of image 3 plus any number so it should not be a problem to get that many number of image. Now for the next time in the loop for a different octave level, the image is recalculated by using the `resize()` from cv2 by passing in the target image as octave base, x dimension as octave base 2nd item shape value by 2 and y dimension as octave base 1st item shape value by 2 and applying interpolation based on the nearest neighbor. This gives the scale-space images in each octave with applied gaussian blur.

```
def generateGaussianImages(image, num_octaves, gaussian_kernels):
    logger.debug('Generating Gaussian images...')

    gaussian_images = []

    for octave_index in range(num_octaves):
        gaussian_images_in_octave = []
        gaussian_images_in_octave.append(image)
        for gaussian_kernel in gaussian_kernels[1:]:
            image = GaussianBlur(image, (0, 0), sigmaX=gaussian_kernel,
                                  sigmaY=gaussian_kernel)
            gaussian_images_in_octave.append(image)
        gaussian_images.append(gaussian_images_in_octave)
        octave_base = gaussian_images_in_octave[-3]
        image = resize(octave_base, (int(octave_base.shape[1] / 2),
                                    int(octave_base.shape[0] / 2)),
                      interpolation=INTER_NEAREST)
```

```
    return array(gaussian_images, dtype=object)
```

As the name suggest it zipped the variable together. Iterable object can be used with this function like array, list, string, dictionary etc. This function pairs the first value of the every iterable object and proceed ahead like this only. So it loop through the gaussian images in octave for each first and second image, the Difference of Gaussian is found out by simply subtracting first and second item of each zipped dictionary.

```
def generateDoGImages(gaussian_images):
    logger.debug('Generating Difference-of-Gaussian images...')
    dog_images = []

    for gaussian_images_in_octave in gaussian_images:
        dog_images_in_octave = []
        for first_image, second_image in zip(gaussian_images_in_octave,
                                             gaussian_images_in_octave[1:]):
            dog_images_in_octave.append(subtract(second_image, first_image))
        dog_images.append(dog_images_in_octave)

    return array(dog_images, dtype=object)
```

From the given image pyramid, it is needed to find the pixel position of all the scale space extrema, meaning this process involves in transforming image data from each octave image or newly formed DoG images into scale-invariant coordinates relative to local features.

An important property of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations. A typical image with a resolution of 500x500 pixels will yield around 2000 stable features (although this number depends on both image content and choices for various parameters). This module requires involvement of knowing the if the image data (pixel) is an extrema or not which is done using our **isPixelAnExtremum** module.

To localize the pixel in the image, firstly it is needed to refine the pixel position in a scale space so, that the features are responsive even if the scale of the image is different.

For this, it is needed to have a regression to the number of attempts as shown in module **LocalizeExtremumViaQuadraticFit**, the number of attempts to try until it hits a convergence is best till 5 because, this process involves in going forward for a lot of DoG images for each data pixel which is an extremum, so exponentially 5.

Each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below to detect the local maxima $D(x, y)$. But firstly the pixel cube is needed to calculate the gradient and hessian values. As the image is defaulted to be in uint8 from it is also needed to convert the pixel cube to float32 also divided by 255 to the pixel cube.

Central difference formula of order $O(h^2)$, where h is the Step size used to find the approximate gradient. Here it is assumed h to be 1, in order to simplify the function to compute a faster gradient at center point.

`lstsq()` is the numpy function where it has 3 parameters to result in least square solution for 3 points where:

- First parameter(array): “Coefficient” matrix.
- Second parameter(array): Ordinate or “dependent variable” values. If b is two-dimensional, the least-squares solution is calculated for each of the K columns of b .
- Third parameter (float, optional) : Cut-off ratio for small singular values of a . For the purposes of rank determination, singular values are treated as zero if they are smaller than $rcond$ times the largest singular value of a .

The result from this gives the extremum update where extremum value is already determined for each image pixel from the pixel cube. If all the absolute value of each result is less than 0.5 then it converges and breaks the loop, else the image index is changed to round integer value from last pixel point extremum. Also, it is needed to make sure the new pixel cube lies within the image, else the result is thrown as extremum moved outside of image before reaching convergence or if the attempt value is more than the estimated attempt value.

Function value is calculated from the current pixel cube extremum which is a dot product using dot() from numpy with the pixel cube [1,1,1] plus 0.5 for convergence saturation, from this it is found the function value at the updated extremum. If the absolute of the function value times the interval value is more than our contrast value i.e.(0.4), then it can be calculated the keypoint for scale space for the updated extremum pixels using the keypoint object creator from **openCv** using hessian trace and dot tracking.

The coordinates of the descriptor and gradient orientations are rotated relative to the keypoint orientation to achieve orientation invariance. A Gaussian function is used to weight the magnitudes, with sigma equal to one-half the width of the descriptor window. The descriptor is then transformed into a vector containing all of the values of these histograms. But firstly, it is needed to scale using scale factor, radius (factor to scale), empty raw histogram using 36 number bin method. it is needed to get value for all the item in the raw histogram by going from negative radius value to current radius value for the images. The image is then broken into regions (x,y) for each current radius in order to get a gradient magnitude and gradient orientation using the difference of gaussian images on x axis and difference of gaussian images on y axis. The weight for each radii and sub-raddi i.e. i and j respectively along with the generic weight factor of the function. Histogram index of the result from the round value of gradient orientation times the number of bins by 360 deg gives us the value of each index of our raw histogram if the region on the initial loop is more than 0 but is less than the image shape.

In order to get the correct feature (keypoint), it is needed to remove all the duplicate keypoints so that it doesn't get any redundant keypoint. Different keypoint's properties are compared such as different points, size, angle, response, octave, class_{i,d}.

First it is needed to convert the calculated keypoint of the image to it's normal size. For this, the point of the keypoint are listed in a tuple while increasing it's magnitude by 0.5, size by 0.5, octave with bitwise or calculation on current octave value and 255.

Making a 3D Model Using LiDAR Technology Like Polycam:

Polycam is the most popular 3D capture app for the iPhone and iPad! With any iPhone or iPad, it can be created high-quality 3D models from photos, and the LiDAR sensor

can quickly generate scans of spaces. 3D captures can be edited directly on the device and export them in a variety of file formats. In this case, the 3D model of the shoe which is the target to be of .obj file, because it is widely used and is supported by the vuforia system.

Step 1 : Taking images of the target from all angle in a 270 deg space for making 3D mesh

LiDAR, or light detection and ranging, is a popular remote sensing method for determining the precise distance of an object on Earth's surface. It is important to know more about LiDAR mapping technology and how it works now that its application has expanded into so many fields.

Using polycam, several pictures are taken of the target in the same background (for consistency) from different angles.

Step 2 : Making 3D object by compiling images into a 3D mesh(3D points) processing. Light detection and ranging (lidar) data are collected from aircraft using sensors that detect the reflections of a pulsed laser beam.

Step 3 : Getting 3D model as Obj file

After the 3D model is given into our polycam from all the images, it needs to be exported it into obj format to use it on vuforia and more. 3D model(mesh) has been made after processing from the compilation server.

Generating Licence Key For Using Vuforia Engine:

To use the vuforia engine, it is needed to generate a unique license key. This supports security within the implementation of vuforia technology.

Vuforia has a developer portal where, it is needed to sign in and generate the license key from the license management tab.

Vuforia has a developer portal where, it is needed to sign in and generate our license key from the license management tab. Inside the License view page, the license key is kept for the database access. This value is used when trying to connect to the vuforia engine on the cloud.

Generating Model Target Database From Vuforia Image Target Generator:

To generate the dataset for unity, a special tool is needed from vuforia that helps to generate data points from the 3D object, target model is generated from this process.

It allows to quickly convert an existing 3D model into a Vuforia Engine dataset. This dataset can be used to create either a single view Model Target, or one that offers automatic recognition and supports multiple models and views.

MTG enables to confirm whether the features of the model will be useable, to set-up the initial snapping position, and then export the final dataset. There is also the option to train your dataset through our deep learning framework for automatic recognition. The Model Target Generator supports popular formats including .obj, .fbx, .pvz, .stl, .igs, .dae, .stp, and .vrml.

Step 1: Create new model target

Firstly,it is the uttermost priority to create a new model target, this requires us to enter the 3D model(obj).Comprehensive 3D model is needed to generate the feature keypoints for further detection.

Step 2: Select model unit

On this process,the unit of the assumed model target is selected. This helps the image target to be detected from a specific scale and width initially. This is important because it involves in the detection of feature point within the image or object.

Step 3: Select model target type

It is needed to select the type of model target, mainly this tell the vuforia generator to generate the system for the targeting mechanism, may it be static (not moving) or dynamic (moving). **Step 4:** Creating viewport from guided view:

A viewport is an angle of the model target to find the features of the model. A viewport helps the model target to be detected and tracked from that point.

Step 5: Training Dataset along with viewport

Using unity and vuforia data package for 3D model placement and application generation

The data package from vuforia using the Model Target Generator, gives the unity compatible package. Using this package is simple, this date is important in feature detection of the target shoe. Firstly, the unity 3d is used to integrate the vuforia feature detection into the system. Unity grants the ability to use a placement tool for 3D model and place vuforia's camera. Placing the 3D shoe into the model target by using the camera is the main technique for real time detection from native camera. Unity has a exporting mechanism for extracting the system to different native applications such as android, IOS, and more.

Finally, to make this system work, a website is introduced where the homepage has a product catalog and on each individual product, there is a try it out button which when the user clicks, an application is downloaded or redirected to app store for downloading the application. By downloading, user is able to augment the 3D shoe and buy the product.

5.2 Testing

Augmented reality apps combine virtual material with the actual environment, allowing for real-time interaction, and they have their own set of characteristics, such as lighting, sensor use, and user position. These applications differ significantly from traditional mouse-and-keyboard programs, necessitating a usability assessment to ensure that they meet their objectives and please users. A realistic criteria is created to assess the usability of augmented reality applications.

5.2.1 Unit Testing

The purpose of unit testing in our system is to determine whether or not every component of our code is functioning correctly and as planned. During the early phase, a number of error is discovered and resolved.

Test Case 1

Objective of test case 1 is to check whether the image is being captured or not. Video feed is given using the footwear application which is captured from the machine's camera. The expected output is get a live feed from the camera.

Table 5.2.1: Unit Testing - Case 1

Table Name	Unit Test
Objective	Capture the live feed from the camera properly
Input	Video feed from the machine's camera
Expected Output	video stream
Original Output	Black screen appears when opening the app
Error info	Attribute Error: something is using the camera in the system

Solution: Closing all the other camera application on the system.

Table 5.2.2: Unit Testing - Case 1

Table Name	Unit Test
Objective	Capture the live feed from the camera properly
Input	Video feed from the machine's camera
Expected Output	video stream
Original Output	video stream
Error info	-

Test Case 2

Objective of test case 2 is to check whether the detection is happening using the model target database. Input given is the video feed of the person's shoe whose record exists in the model target database. Expected output is the 3D model of another shoe is occluded into the video feed.

Table 5.2.3: Unit Testing - Case 2

Table Name	Unit Test
Objective	Occluding the 3D model of a shoe
Input	Video feed from mobile's camera
Expected Output	The shoe is occluded into 3D model of another shoe
Original Output	The shoe is not occluded.
Error info	viewport not matching

Solution:

The problem is that the viewport is the initial way of detecting the feature's of the target. So, the viewport is adjusted to properly detect the model target.

Table 5.2.4: Unit Testing - Case 2

Table Name	Unit Test
Objective	Occluding the 3D model of a shoe
Input	Video feed from mobile's camera
Expected Output	The shoe is occluded into 3D model of another shoe
Original Output	The shoe is occluded into 3D model of another shoe
Error info	-

5.2.2 Integration Testing

Table 5.2.5: Integration Testing - Case 1

Table Name	Integration Test
Objective	Augmenting the 3D model of a shoe
Input	Video feed from mobile's camera
Expected Output	The shoe is Augmented into 3D model of another shoe
Original Output	
Error info	The shoe is not augmented

Solution:

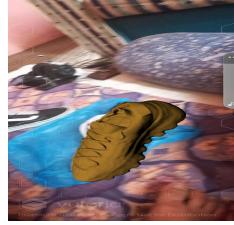
When the image was scanned by the AR application, it initially didn't augment 3D model to it. When the cause of the error was properly identified, it was removed and again re-tested, then the result was properly displayed.

Table 5.2.6: Integration Testing - Case 2

Table Name	Integration Test
Objective	Augmenting the 3D model of a shoe
Input	Video feed from mobile's camera
Expected Output	The shoe is Augmented into 3D model of another shoe
Original Output	
Error info	-

5.2.3 System Testing

Table 5.2.7: Test Case: Without Wearing Shoes

No.	Description	Input	Expected Output	Actual Output	Status
1	Open Application and point on the shoe from front	Image of shoe from front	Successful Augmentation		Pass
2	Open Application and point on the shoe from right	Image of shoe from right	Successful Augmentation		Pass
3	Open Application and point on the shoe from left	Image of shoe from left	Successful Augmentation		Pass

Continued on next page

Table 5.2.8: Test Case: With Wearing Shoes

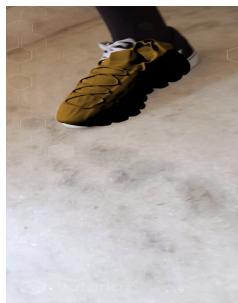
No.	Description	Input	Expected Output	Actual Output	Status
1	Open Application and point on the shoe from front	Image of shoe from front	Successful Augmentation		Pass
2	Open Application and point on the shoe from right	Image of shoe from right	Successful Augmentation		Pass
3	Open Application and point on the shoe from left	Image of shoe from left	Successful Augmentation		Pass

Table 5.2.9: Test Case: With Another Shoe

No.	Description	Input	Expected Output	Actual Output	Status
1	Open Application and point to the shoe at zoom level	Image of the shoe	Successful Augmentation		Fail
2	Open Application and point on the shoe from normal range	Image of shoe from normal range	Successful Augmentation		Pass
3	Open Application and point on the shoe from left	Image of another shoe from left	Successful Augmentation		Pass

5.3 Result Analysis

The AR-based try on tech was manually tested with different set of data and it performed as expected in most of the cases. All functional requirements were also met during the testing procedure. Each module of the system was tested to check the correctness of the output. The expected result was compared with the outcome obtained after performing various changes. For testing, we collected the sample images for 3 test cases.

Case 1 :

In this case, three images without wearing a shoe were collected among which all 3 images were successfully augmented by 3D shoe model. So, the success rate for case 1 is :

$$3/3 * 100\% = 100\%$$

Case 2 :

In this case, three images with wearing a shoe were collected among which 3 images were successfully augmented by 3D shoe model. So, the success rate for case 2 is :

$$3/3 * 100\% = 100\%$$

Case 3 :

In this case, three images of a different shoe were collected and 2 out of 3 images were augmented by 3D shoe model. So, the success rate for case 3 is :

$$2/3 * 100\% = 66.67\%$$

Total Success Rate

The total success rate of the system can be calculated as :

$$100 + 100 + 66.67/3\% = 88.89\%$$

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

The purpose of the application is to augment the 3D shoe model to the foot of the person. After the completion of the project, the purpose was fulfilled of augmenting 3D shoe model to the foot. With the help of different tools and technologies, an AR shoe tryon application developed to provide customers with deeper and more comprehensive information about their products from home.

6.2 Future Recommendations

This application can be extended further to include many other shoe model targets. Due to the time constraint, this application only used one shoe model target and only one feet was used to demonstrate the augmentation of 3D shoe model to the feet.

The accuracy can also be further enhanced by better placement of model target and test model. Also when users want to try out new shoes, each time it is needed to download a separate application which is a severe limitation which couldn't be solved due to time constraints. Other apparels such as slippers,crocs can also be included.

References

- [1] Kaiying Cao et al. “Whether a retailer should enter an e-commerce platform taking into account consumer returns”. In: *International Transactions in Operational Research* 27.6 (2020), pp. 2878–2898.
- [2] Yuzhao Liu et al. “Comparing VR-and AR-Based Try-On Systems Using Personalized Avatars”. In: *Electronics* 9.11 (2020), p. 1814.
- [3] Gustav Amer. “Foot Detection Method for Footwear Augmented Reality Applications”. In: *Semantic Scholar* (2016).
- [4] Abdullah Khan. *Foot Tracking byDeformable Line Templates in Smart phones*. 2012.
- [5] Qing Tan, William Chang, et al. “Location-Based Augmented Reality for Mobile Learning: Algorithm, System, and Implementation.” In: *Electronic Journal of e-Learning* 13.2 (2015), pp. 138–148.
- [6] Anders Henrysson. “Bringing augmented reality to mobile phones”. PhD thesis. ACM, 2007.
- [7] Anders Henrysson, Mark Billinghurst, and Mark Ollila. “Virtual object manipulation using a mobile phone”. In: *Proceedings of the 2005 international conference on Augmented tele-existence*. 2005, pp. 164–171.

A. Appendix

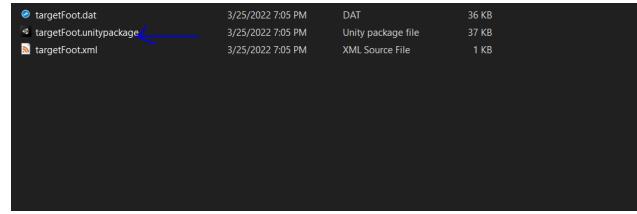


Figure 8.0.1: Vuforia model target data for unity

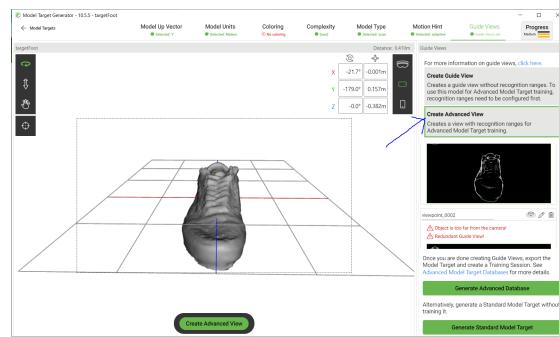


Figure 8.0.2: Creating guided view for target model

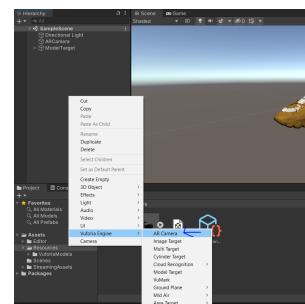


Figure 8.0.3: Creating AR Camera for unity using vuforia package

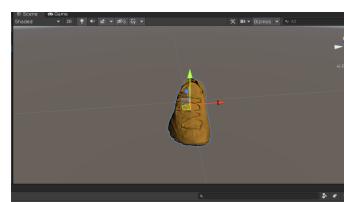


Figure 8.0.4: Replacement model

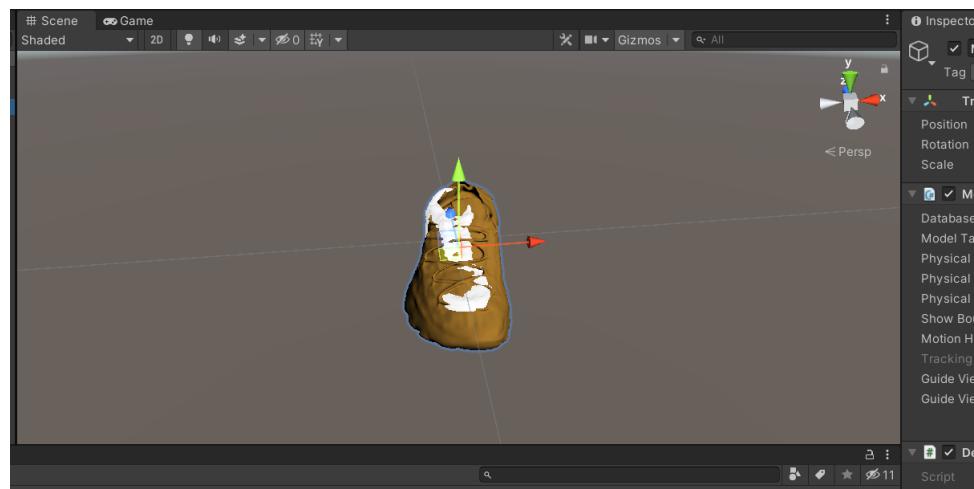


Figure 8.0.5: Placing replacement model on top of target model

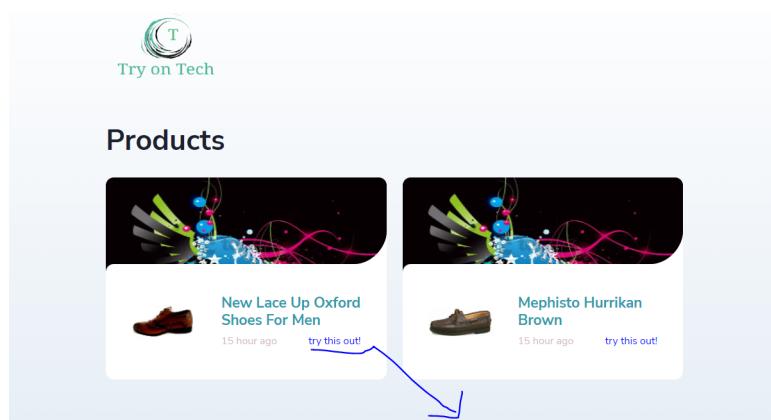


Figure 8.0.6: Try it out button



Figure 8.0.7: Trying out the shoe app