3D AR RECONSTRUCTION

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jośe State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Computer Science

By

Sneh Arvind Kothari

May, 2023

3D AR RECONSTRUCTION

By

Sneh Arvind Kothari

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE SAN JOSÉ STATE UNIVERSITY

MAY 2023

Dr. Christopher Pollett	Department of Computer Science
Dr. Robert Chun	Department of Computer Science
Dr. William Andreopoulos	Department of Computer Science

ABSTRACT

The goal of the project is to improve the shopping experience for users by using augmented reality technology. People generally want opinions from others when buying shoes offline. Clicking and sending images of a shoe is not an ideal solution as it does not give the complete feel of the shoe. We developed the 3D AR Reconstruction app to make this process better. A user of our app clicks photos of the shoe. This image data is converted to form a mesh that can be shared. On receiving a model the user can open it in the app and interact with it to get a better look and feel of the shoe. This can be particularly useful in something like Amazon where sellers can click images to build a mesh and customers can render this mesh for a better buying experience.

The project is implemented by having a phone app to click images and view the model in augmented reality. It also has a server that converts 2D images to 3D mesh. There are existing apps that allow you to view meshes in 3D and convert images to 3D models but there is no app that allows you to build 3D models from images and view those models in 3D. We did user testing which gave us great insights in building the project and improving the project in the future.

ACKNOWLEDGEMENT

I want to express my gratefulness to my project advisor Dr. Christopher Pollett for the continuous guidance and support during the creation of this master's project over the past year. I am heavily indebted to him for the skills and knowledge gained while working with him. I would also like to express my appreciation to other faculties in Department of Computer Science at San José State University for their efforts and guidance while teaching the advanced computer science courses. Lastly, I would like to thank my family and friends for their constant care and encouragement during my journey for completing Master of Science in Computer Science degree.

TABLE OF CONTENTS

TABLE Of FIGURES

I. INTRODUCTION

Augmented Reality is the integration of digital information with the user's environment in real-time. AR users experience a real-world environment with generated perceptual information overlaid on top of it. AR applications are growing in popularity and are being adopted by developers in their products. Companies like Snapchat, Instagram have incorporated this and have placed things like caps and glasses on users faces. Another product that has been continuously gaining more and more attention from people is sneakers. The sneaker industry has grown from 45 billion dollars in 2014 to 72.2 billion dollars now and is expected to be north of 100 billion dollars by 2030. It has become a style statement for people. This project is aimed to cater to people from these two groups. Imagine you go to buy a pair of sneakers and you love them but want a second opinion. The normal way is to click a photo of the sneaker and send it to people. The issue with this method is that a 2D image does not capture the sneaker correctly. Hence the opinion you receive is not accurate. This problem escalates even further when you are shopping for someone. If you are buying sneakers for someone they would want multiple images and then try to figure out how the sneaker will look. It would still be difficult to visualize the complete sneaker. Hence to solve this problem we are building 3D AR Reconstruction app.

The basic idea of the 3D AR Reconstruction app is that when you are in a store you can scan the shoe by clicking multiple images of the shoe. Following this, we build a 3D model of the shoe and provide the user with a 3D model of the shoe which the user can view in the app. The user can also share the 3D model with other users of the app. This can be done by sharing the deep link of the model. Deep linking is sending the user data to a particular location in the app rather than the generic app home. Any user that has access to this deep link can view the model in the app. The model can be viewed by placing it in the real world. So the user can point and place the object on a plane and interact with it to get the proper feel of the shoe and make a better more informed decision when shopping.

Another use case for this app is in the eCommerce space. So this can be an offering from a company like Amazon to its sellers where the sellers can give a better feel to their customers. Sellers can build a 3D model of their shoes by scanning them and then allowing customers to see them in 3D and even try them on. This can be a real differentiator in generating more sales for the sellers. Ikea does the same for furniture it sells. It allows users to place the furniture in their home so they can decide before buying.

Currently, there are some companies actively working in this domain. Wanna Fashion which is developing AR products for sneakers and watches. The key difference between our idea and their app is that they rely on 3D models provided by companies or preexisting models. We are currently working on making an app that allows you to make and share models. Amazon also recently announced its entry into virtual shoe try on space. Amazon partnered with Reebok and Adidas so far.[12] This indicates that the sellers need to provide the models for now. Amazon has claimed they will be expanding to more sellers which also means that they are working on the mesh generation technology and there should be some tool or research available soon. Snapchat is also a major player in this domain where they have built a suite of APIs and models that allow developers to use their technology to track objects and movements and overlay mesh on them.[13] This seems like a great tool to improve user experience and making interaction and tracking better.

The report has four major sections describing the project and the system built during the course of this project. The first section is preliminaries which talks about the exploratory apps that helped in getting all the kinks out and plan the project. Following this is the design section where we talk about the main components of the app and how these sections interact with each other. Following that is an implementation section, which goes into detail regarding how the project is more details about the components. There is a technical challenges section where we talk about issues faced during building this project followed by testing and results section which outline the different tests and results. Finally w end the report with a conclusion.

II. PRELIMINARIES

This section discusses the basic concepts necessary to familiarize with the different aspects of building an 3D AR Reconstruction app. In this section we first explore the method of building a basic AR app that places teapots on tapping on the screen. We then move to building the 3D assets, i.e., how to build a 3D model from 2D images. Following which we explore a way to share data between users and accessing it via the app. Lastly, we experiment with placing shoes on the screen and allowing user to interact with them. These experiments serve as a way of exploring different methods of building the system and serve as building blocks of the final system.

A. AR App with teapots – Deliverable 1

The goal for this deliverable was to build a basic AR app. This deliverable was aimed at getting familiar with AR frameworks and also deciding on what technologies, frameworks, and tools to use in building the 3D AR app. It was also aimed at understanding the different types of interactions that we can have with an AR object.

In the weeks leading up to Deliverable 1, we read a paper that was a comparative study between ARKit and ARCore. This helped me in understanding the pros and cons of these two frameworks. Following this, we spent a week reading about ARKit and all the frameworks that make up the AR experience in an iPhone.

In the first week of the deliverable, we further explored the frameworks that make up the AR experience. We came across RealityKit which uses information provided by the ARKit framework to integrate virtual objects into the real world. This framework allows importing fully formed meshes into the real world which we will be using for our teapots in this deliverable. It also supports interactions (translation, rotation, etc.) on the object and can allow changes based on environmental changes. We also came across other frameworks which could be used in place of RealityKit but we finally settled for RealityKit because it was the latest framework and the most powerful and optimized one. One more framework that showed promise was Metal but it has a steep learning curve hence we stuck to RealityKit.

The app uses a mixture of UIKit and SwiftUI to build the UI of the app. The app also has a 3D model of a teapot. Once you open the app and tap on New Room. It first asks permission for your camera and then you can view the room from you on the screen. Now when you tap on the floor you place the pot there. This was done by recognizing the tap gesture and if that is a horizontal plane then anchoring the teapot to the ground at that location. Every time you tap on the screen you place a pot on the screen. The app also allowed you to move virtual objects. This app shows some of the capabilities of the AR frameworks and will be used in the project when building AR Shoe Reconstruction.

This was the first AR app we built and it served in helping in understand the basics of building an AR app. This app also helped in putting light to problems that we will encounter in the future. Like when you place a pot on the screen it freezes for a second or how will multiple entities interact when placed one on top of the other. This served as a great starting point for us. In the final app we introduce interactions with the entities. We also build the model from images rather than using premade models available from the internet.

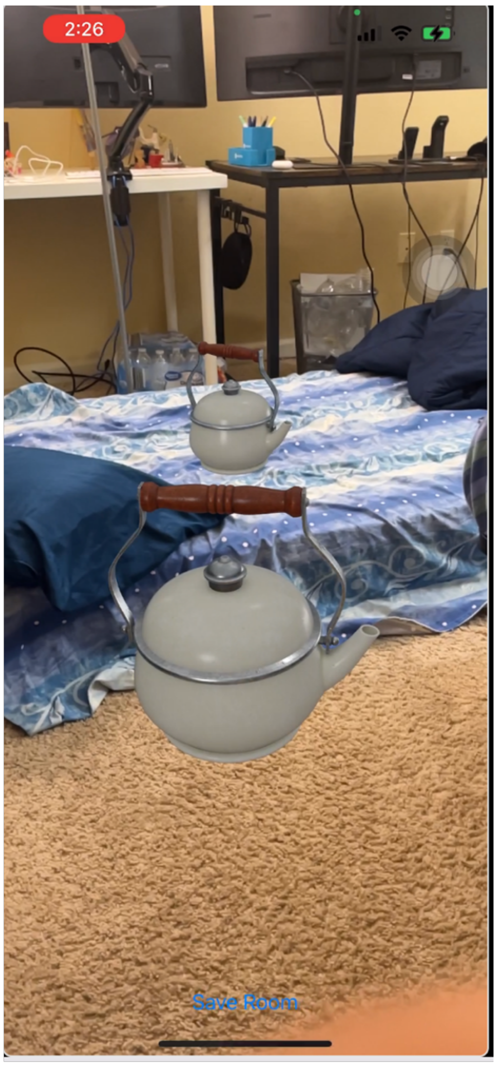


Figure 1: Demo screenshot

B. Code to make 3D mesh from 2D image data – Deliverable 2

The goal of this deliverable was to explore a way to build a 3D mesh. It was also aimed at exploring the way apps can capture data from different sensors on the phone and how we can use that sensor data to build a mesh.

The initial idea for building the app was to use the Lidar sensor on the phone and get depth information using that. Finally, this sensor data can be used to build our 3D mesh. There were blogs about scanning entire rooms using the Lidar sensor. So the idea was to scan it like a room and crop out the object and use it in our app. The issue with this is that the sensor is only available on Pro models of the iPhone. Hence, this idea was dropped due to the lack of access to a Pro phone and which would limit people from using it.

While looking for other ways to make a 3D mesh from 2D images we also found that Apple provides the Photometry API. This API accepts 2D images and converts those images to 3D models. This seemed like a reasonable idea for the deliverable but has a lot of shortfalls. So for Deliverable 2, there was a lot of time spent on what works for our case. For Deliverable 2 the code was mostly from Apple's sample project on creating a 3D model from 2D images.

This deliverable has two apps, one app for the phone where the user clicks multiple photos of the object from the app and second for the laptop to build model from those images. The images needed to be clicked using the app because the app utilizes the dual camera on the iPhone and provides a depth and a gravity file both of which are needed to build a 3D model. We click multiple images of the object using this app. In the sample example provided by Apple, they needed to click more than 300 images to create a mesh of a pot. So for our case of the ball, we clicked more than 200 images to build the mesh.

After clicking these images we shared all the images with their supporting files to the laptop. We do this because the Photometry API is only available on the MacBook hence we need a second app on the laptop. These images were then used as input for the MacBook app and the photometry app built a 3D mesh out of this. This worked for our case where we clicked more than 200 images for a basketball.

The issue with using this system was that the Photometry API only works on a MacBook hence for our actual use case we will need to have a server with macOS and XCode so that we can make the 3D model. This did not seem very reliable. Another issue was that the system needs a lot of images to make a 3D model. For a simple ball, the system failed to work when we had 100 images.

Hence, we realized that we need to write our 2D images into 3D model building code. Initially, the idea was to have everything run on the phone so we thought of running the model building code on the phone. The issue with this is that we could not find new papers on implementing this on phones and the old ones we found were using phones older than iPhone X and did a bad job of making 3D meshes and were very slow. Since the phones are more powerful now it might still work for us, but it would have to be run on the phone's GPU which would require code to be written in Metal(steep learning curve).

So we decided to try the other idea which is to run the 2D image to 3D model code on a server. This way we can do the heavy work on the server with more processing power. Since this is run on a server we can use Open Source frameworks and libraries built for this purpose. Also, we did not have to be restricted by Metal or any language.

So for the final project, we get image data from the phone, then upload that data to the server where we do image stitching and build the model. This model is then accessed by the app.

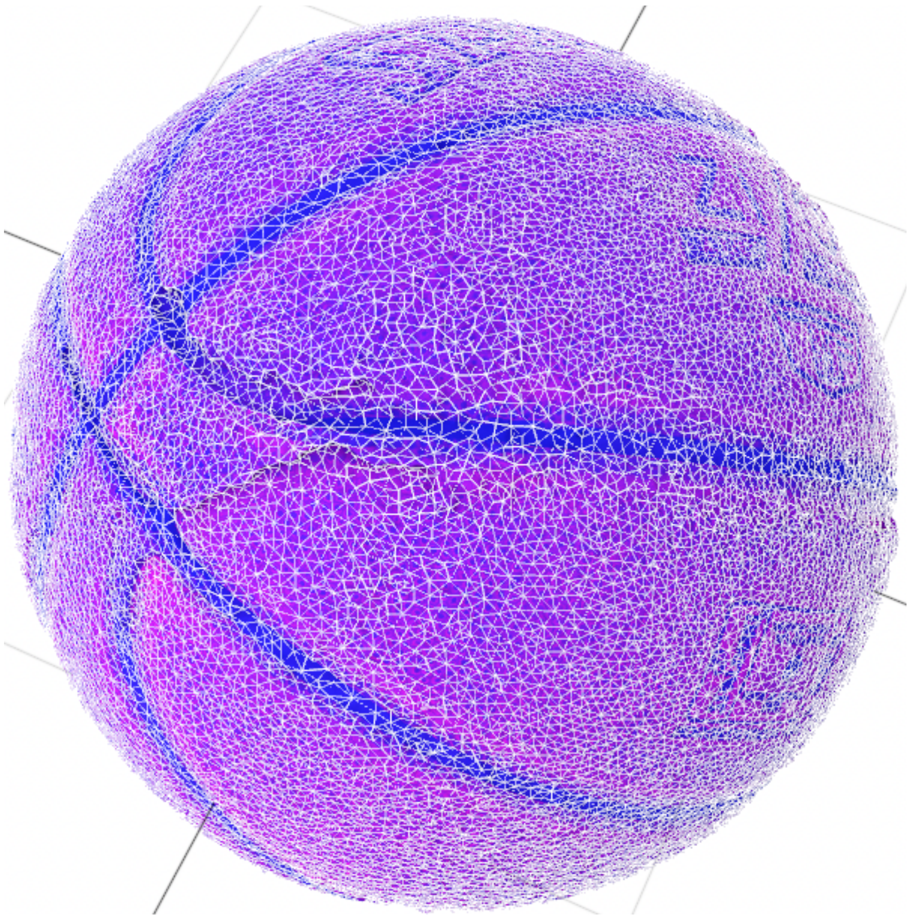


Figure 2: Ball 3D model

C. Exploring Sharing Options – Deliverable 3

The goal of this deliverable was to explore different ways to share the 3D model. Trying different ways to share the 3D model and since now we will be getting the model from a server are there any more ways to share the 3D object?

The initial idea when starting this was to allow 3D object sharing via iMessage. The issue with this idea is that Apple provides a custom option to view it using its API which does not give a great personalized experience. Another issue is that this object is stored on the user's iCloud and this will lead to insufficient space. One more issue that can happen is that iMessage limits the size of messages hence we did not face issues in our tests but it might happen.

Since we are already integrating with the server for making a 3D model out of 2D images. It was intuitive to use it for model sharing. So we finally decided to upload the 3D model to an S3 bucket which can be accessed by other users using the object's id.

The sharing problem remains in this case where we get the S3 link for the object. So we implemented deep linking which is we gave our app a unique identifier and when the object will be shared it will embed the S3 id in this link which will be used by anyone with this link to download the 3D model and view it in the app.

Since the goal of this deliverable was to explore sharing we implemented a basic version where based on the id passed to the app a unique URL was generated which led to a random image being downloaded. This was a way to show that we can deep link apps and based on the id fetch data. S3 is the sharing option we will be using for our final project.

D. AR app to render shoe model on screen – Deliverable 4

The goal for this deliverable was to explore different user experiences while interacting with the shoe model. It was basically to decide the optimal way to show the shoe on the screen. There were two ideas for how to implement this deliverable.

The first idea is that when the app is opened and the shared shoe needs to be presented to the user we should fix the shoe at a particular distance from the camera so that however the user moves the camera the shoe is always at a fixed distance. This seemed like a great option but the issue here is that in Apple's ARKit framework every 3D object needs to be anchored to one of the planes. So the shoe has to be anchored on a horizontal or vertical plane which cannot work if it always has to be at a fixed distance from the camera. Hence this idea was dropped.

The second idea was to allow the user to place the shoe on the horizontal plane in front of him by tapping on the plane. This anchors the shoe at that location. Now users can walk around that point to understand how the shoe looks. This also seemed a bit tedious which is why we added rotate gesture to the app, this can be used by holding the shoe with one hand and swipe with the other hand to rotate the shoe and look at it from all angles. The shoe can also be resized and moved for a better view and to look at specific parts of the shoe.

This Deliverable helped in defining the actual look and feel of the AR object that the user will experience. It is the basis of the AR View module of the final project.

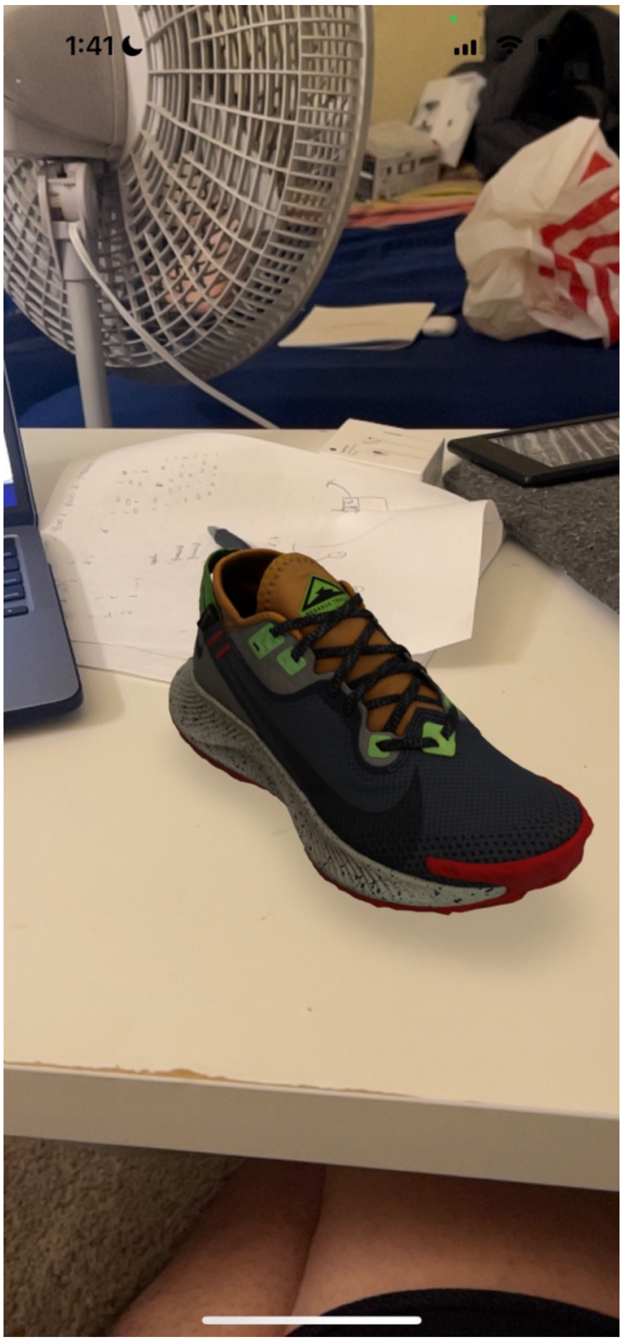


Figure 3: Shoe in AR

III. DESIGN

The objective of this project is to help users to experience shoe shopping in a new and unique way. It is aimed at helping users make more informed and better decisions while shopping for shoes. The project is also aimed at improving the sales of online stores by allowing users to try their shoes on. This project can help in providing small and medium size businesses the same edge that companies like Nike and Gucci have.

This project can be divided into three modules and these three work together to create the user experience. The first part is the image collection part where the user clicks images on the phone app and uploads it to an object store. The second part is the server where the server downloads the images and uses Photogrammetry to convert the images into a 3D model. Photogrammetry is the process of creating a 3D object using a series of images using specialized software. It is done by taking measurements from photos to find the exact position of surface points. Following this, the server uploads the model to the object store. The last part is the model experience. This is where the user can download the model from the object store and interact with it. This helps the user make a better and more informed decision.

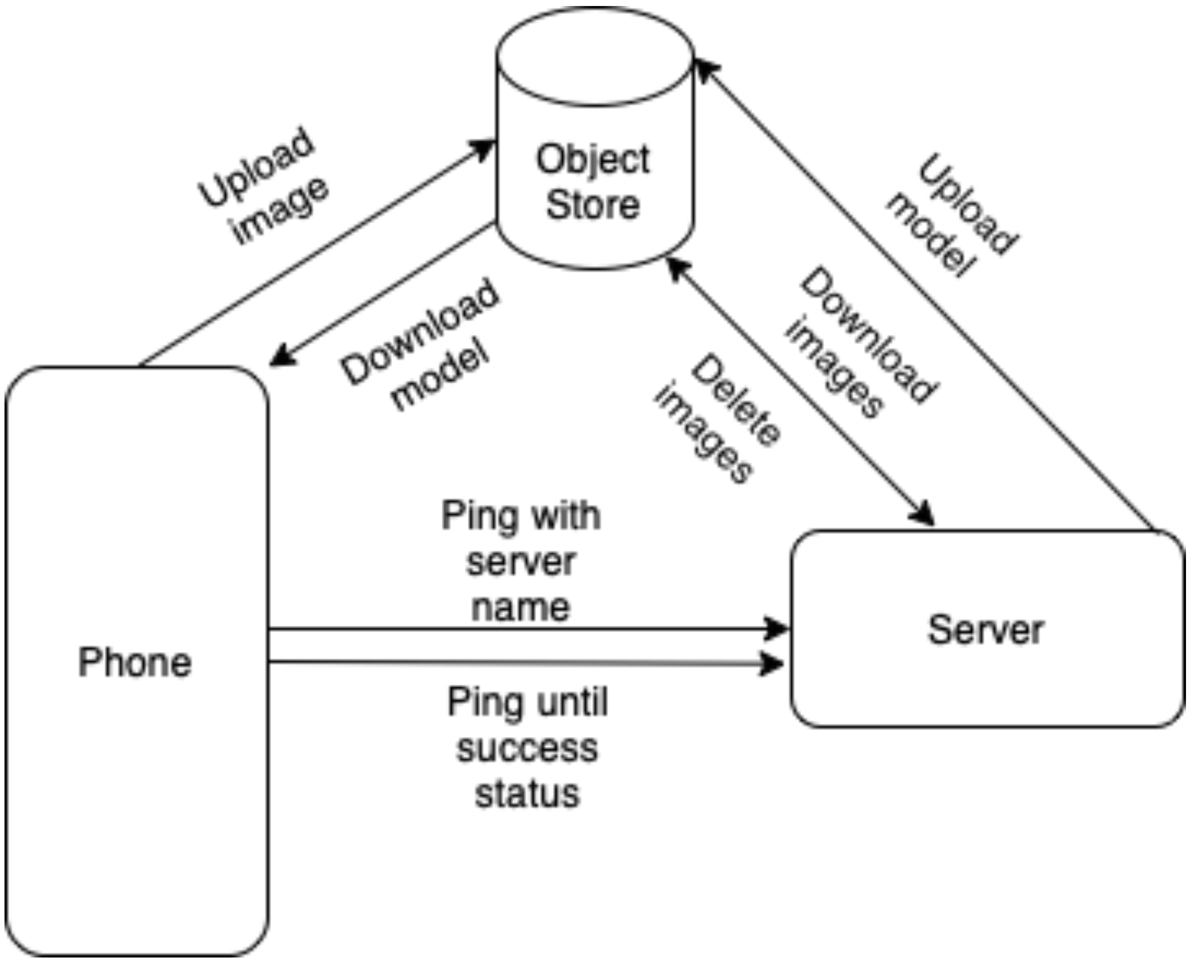


Figure 4: Design of the system

A. App to click images of the shoe

This is the module where the user clicks images of the shoe to build a model out of it. The app takes camera permission and storage permissions to capture and store the images on the device. The device captures images along with the depth data. Capturing depth data is important here because if only images are captured then the number of images have to be more for a good model. This slows down the entire process on the phone side for uploading and the server side for model creation. Hence, depth data is captured. The user keeps on clicking the images following which the user can view images and delete all the images that are not of good quality. Following this, the user clicks the Approve button. The user is then asked for the name of the shoe. Then the phone zips the images into a file and sends it to the object store. After the upload is complete the device pings the server with the zip folder name. Following this, the phone pings the server every 30 seconds to check if the model building is complete. Once the build is complete the device gets a response with the model name. The device then calls the view and interact module(Module C) with the downloaded model.

B. Server to convert 2D images into a 3D model

The server first gets pinged when the device has uploaded the data to the object store. The server receives a hashed value and a filename from the phone. The server uses the hashed value to identify the authenticity of the user and only then proceeds with the next steps. Once the user is verified the server takes the file name and downloads the zipped file from the Amazon S3 bucket which is our object store. The server unzips and stores the file locally. Following this, the server deletes the file on the object store. This is done to avoid having excessive data in S3 buckets which can help in reducing costs. The server then runs the code for converting the 2D image to the 3D model. Once the model is built the server deletes the files locally and uploads the model to the S3 bucket with the same name as provided by the user. During this entire process, the server is continuously pinged by the phone for an update. The server returns the loading state. Once the model has been uploaded to the S3 bucket the server returns a success state.

C. AR App to allow user to view and interact with the shoe

This view uses the camera to allow the user to view AR objects in the real world. The user taps on the screen to place the shoe in front of them on a horizontal plane. The user can interact with the shoe by moving it along the horizontal plane. The user can also rotate the shoe to get a better feel of the product and view it from all angles. This can be done by holding the shoe with one finger and then swiping with the other finger to turn the shoe. Another interaction that is available which is an important aspect of shopping is the zoom option. This can be done by pinching the shoe. Zooming in on the shoe can help users get a better feel and understanding of the shoe.

This module also has a list view which lists all the shoe models downloaded and available to the user. The user can select a model to view. The user can share the model with other users from this screen and also delete the models.

IV. IMPLEMENTATION

The project has two major implementation components. One is the server where 2D images to 3D model rendering happens and the second is the phone app where the images are taken and the AR models can be seen.

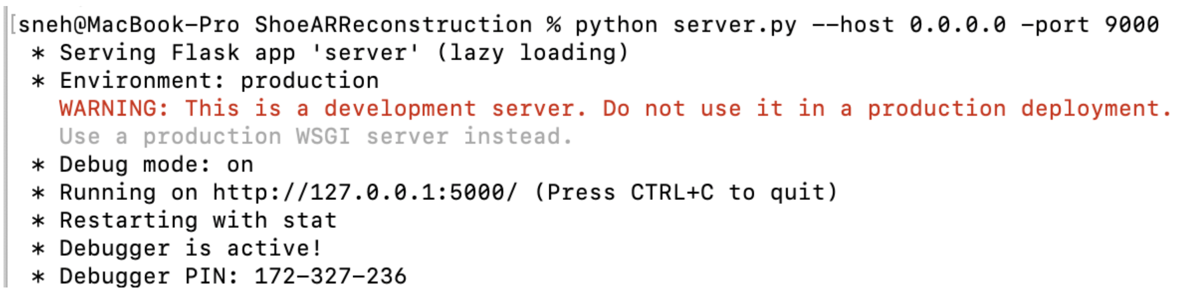
A. Server

Flask Server

Flask is a micro web framework written in Python designed to enable developers to create and scale web applications quickly and easily. The server is responsible for converting the 2D image data to a 3D model. This is a Python Flask server. The server first receives a request from the phone which consists of the filename corresponding to the file on the S3 bucket and a hash value that the server uses to establish authenticity. The Flask server first verifies the user following which the server makes an AWS client using the credentials and then connects to the S3 bucket to download the file.

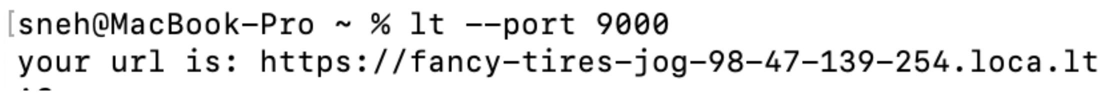
After downloading the server unzips and stores the folder at a particular location. The server then calls a native macOS app to convert the 2D image data to a 3D model. This is an executable file and the server runs this program. The server waits for this process to get over following which the server uploads the file to the S3 bucket with the same name it received from the user.

During this process the phone continuously pings the server for an update and the server responds with a waiting state. Only after the upload is over does the server return success and the phone stops pinging.



```
[sneh@MacBook-Pro ShoeARReconstruction % python server.py --host 0.0.0.0 -port 9000
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 172-327-236
```

Figure 5: Running Server on localhost

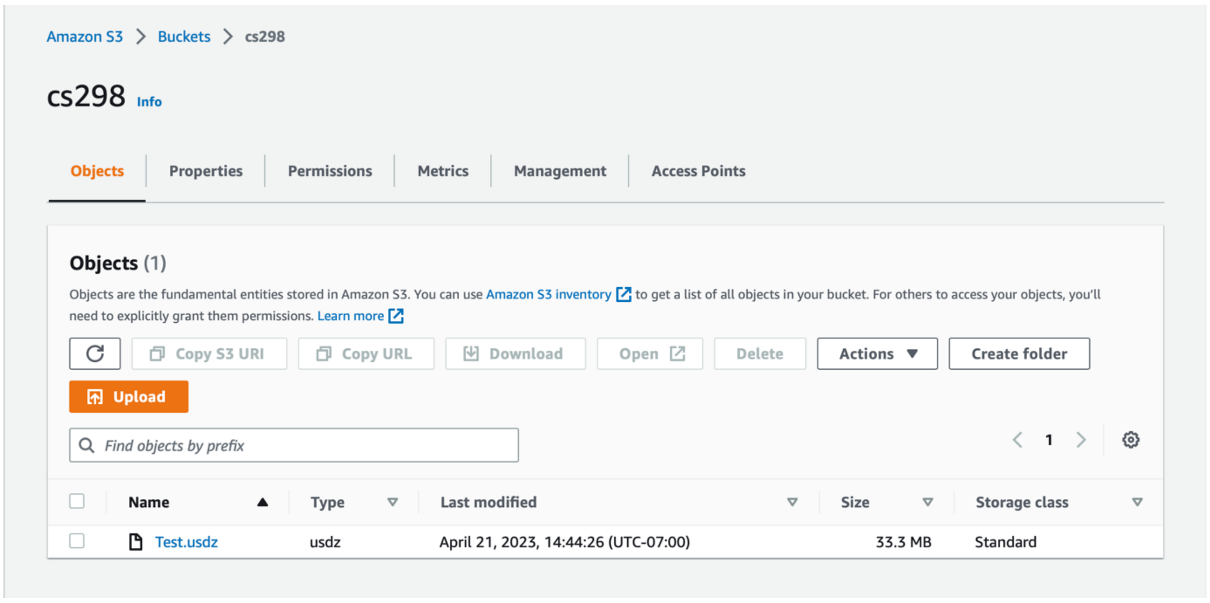


```
[sneh@MacBook-Pro ~ % lt --port 9000
 your url is: https://fancy-tires-jog-98-47-139-254.loca.lt
```

Figure 6: Establishing a tunnel connection from localhost to the above URL

2D to 3D model code

The executable file is a part of the server and is the code for converting the 2D image file to a 3D model. We use the Photogrammetry API provided by Apple to build the model from images. This API is a part of the RealityKit framework which is used for building 3D models and experiences in an Augmented Reality environment. This is a native macOS API and hence we have the server run the executable file. This API takes a bunch of images and configuration as input and outputs the 3D model. Since we are working with sneakers we select the model to be detailed. We do this because some minor details of the sneaker might be missed which might lead to a bad user experience. Since we also expect the user to rotate the phone in one direction we expect the photos to be ordered. This also helps in improving the speed of the model building.



Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Test.usdz	usdz	April 21, 2023, 14:44:26 (UTC-07:00)	33.3 MB	Standard

Figure 7: S3 to upload photos and download models

B. Phone App

On the user side, we built a phone app that interacts with the server and takes user input, and processes images to build a model. We built a native iOS application using Swift and SwiftUI for this purpose.

Image Capture

We call the image-clicking module of the app image capture. This module is responsible for clicking images and then storing them on the phone. In this module, we capture an image along with its depth information and store it on the phone. Privacy permissions are added to the info.plist file for this. Once the user has captured all the images the user can go to the image gallery and delete photos that are not clear. Following this the user taps on Approve.

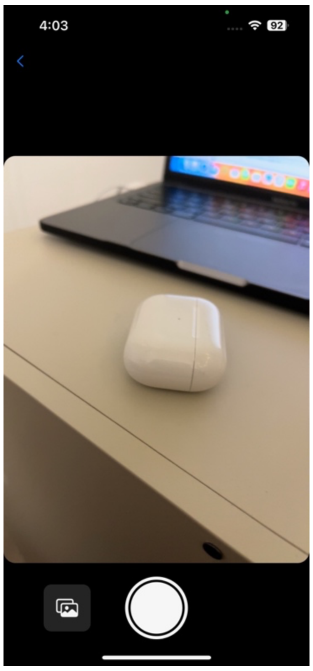
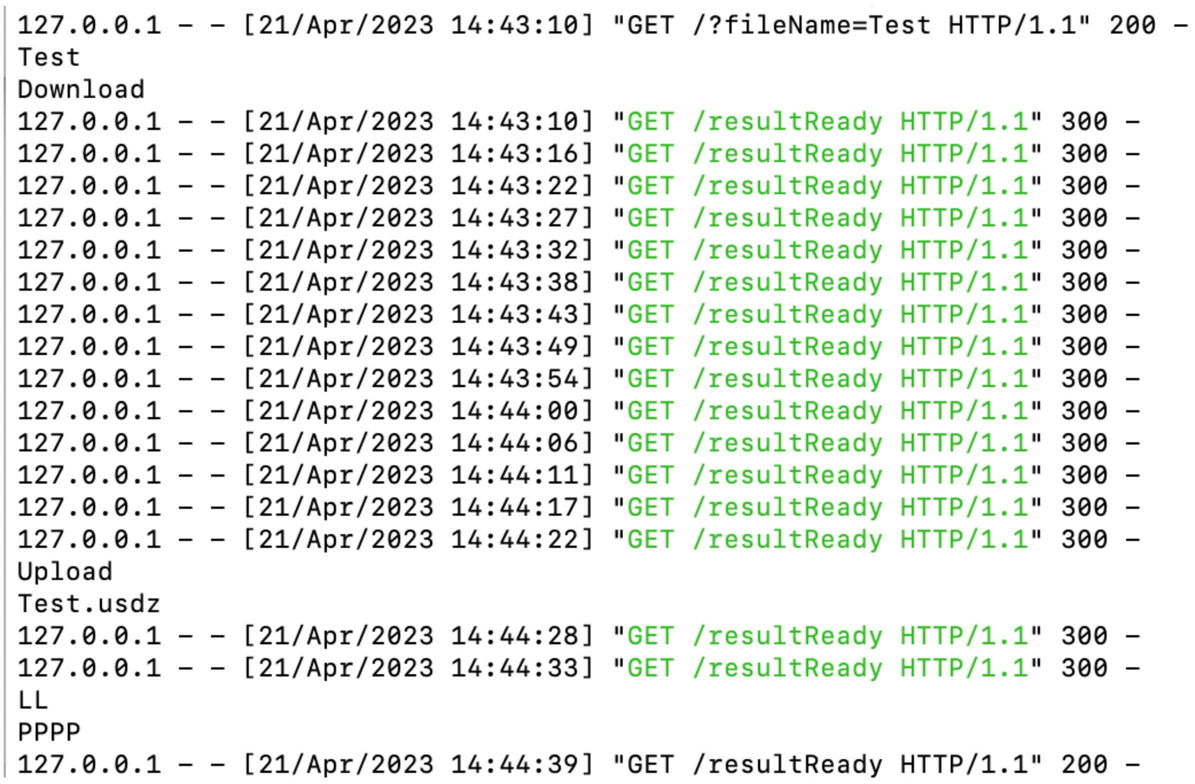


Figure 8: Camera View for Clicking images

Photos and Model Handler

This module is responsible for all the server interactions and communication outside the app as well as the supporting functions to facilitate this. Once the photos are approved this module zips the files and then asks the user for a name. Following this, it establishes a connection with the S3 bucket and uploads the zipped file to the S3 bucket. It then sends the filename along with the verification key to the server. This occurs on one thread and the main screen shows loading. Following this, the phone periodically pings the server to get an update about the model. Once the module receives a success state from the server it downloads the model from the S3 bucket and then triggers the AR View module to view the AR object in the real world.



```
127.0.0.1 - - [21/Apr/2023 14:43:10] "GET /?fileName=Test HTTP/1.1" 200 -
Test
Download
127.0.0.1 - - [21/Apr/2023 14:43:10] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:16] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:22] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:27] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:32] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:38] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:43] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:49] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:43:54] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:00] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:06] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:11] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:17] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:22] "GET /resultReady HTTP/1.1" 300 -
Upload
Test.usdz
127.0.0.1 - - [21/Apr/2023 14:44:28] "GET /resultReady HTTP/1.1" 300 -
127.0.0.1 - - [21/Apr/2023 14:44:33] "GET /resultReady HTTP/1.1" 300 -
LL
PPPP
127.0.0.1 - - [21/Apr/2023 14:44:39] "GET /resultReady HTTP/1.1" 200 -
```

Figure 9: Continuous pings from phone to server

AR View

This module is responsible for allowing the user to view the model in the real world. When the user selects this view the user gets an option to view one of the shoe models already downloaded on the phone. On selecting the model the camera opens up and the user sees the world around them. This module scans the image it receives from the camera to identify the horizontal plane where the user can place the shoe in front of themselves. On tapping the screen the 3D asset which is a shoe is placed in front of the user. The user is allowed to perform certain operations to interact with the shoe. The interactions are added by adding gestures to the object which is the shoe. The gestures are, translate which means moving the shoe around. Pinch which is zooming and hold and move which is rotation.

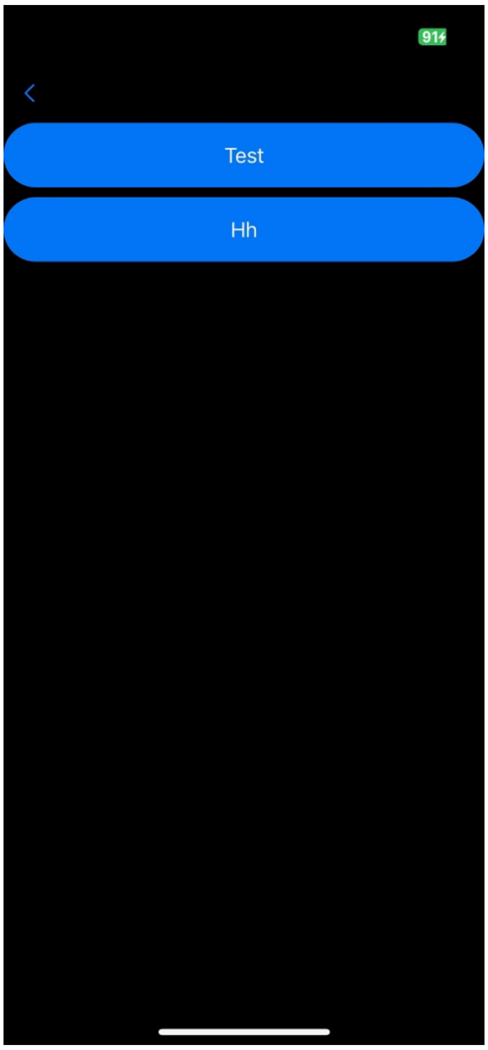


Figure 10: List of shoe models available

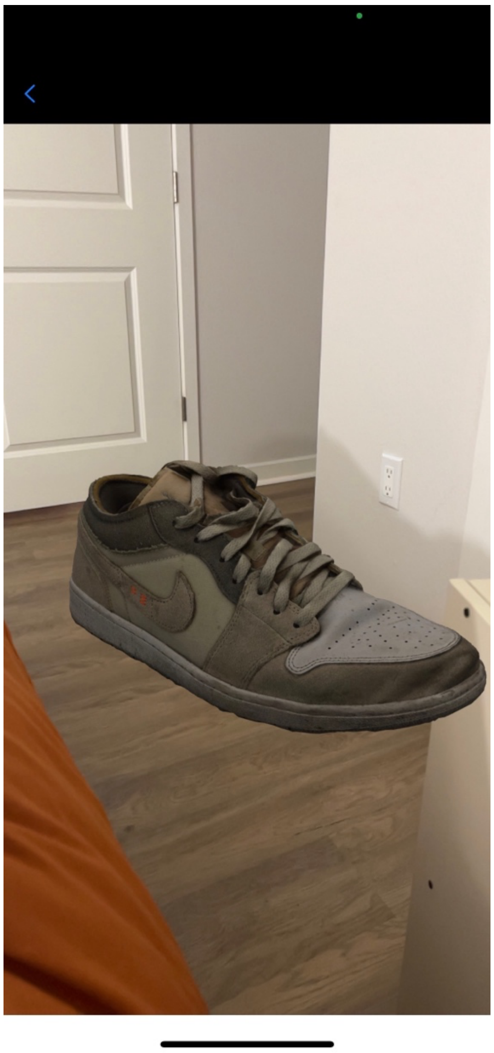


Figure 11: AR View of the shoe entity

APP Walkthrough

We first open the app and see the options view. Options View provides 2 options. Either the user can go to the Camera View which is the image-clicking part of the app or the AR View which is the AR experience of the app. If the user clicks Camera View then the camera is opened and the user can start clicking images. Once the user is satisfied with the number of images the user can go to the gallery view and tap on Approve. On tapping Approve the app enters the loading state and the photos and Model Handler module is called which uploads the photos and then pings the server and then downloads the model. Following this, the app navigates to the AR View, and the user can place the model in front of him.

In the scenario where a user navigated to the AR View the user would see a list of shoe models available. The user can then long press on one of these models to share them with others. The user can also delete the models. The user can tap on the model which opens the camera and then the user can place the selected model in front of them.

VI. TESTING

We perform user testing on our application and observe users how users interact with the application. We recorded their response and have divided and tabulated them into 3 tables and feedback. While testing with the users we recorded the number of images captured and if depth data was captured along with it or not. All these tests were performed in the same environment which is completely white background and with same lighting conditions and avoiding all reflective surfaces. We also placed lights at angles in order to minimize shadows. These were the observations from the tests we performed.

Test 1: Reconstruction quality with varying number of images

User Number	Number of Images	Depth Data captured	Reconstruction Quality
1	5	Yes	Poor
2	15	No	Poor
3	60	Yes	Excellent
4	102	Yes	Excellent
5	20	Yes	Good

Table 1: Reconstruction Quality with varying number of images

We have called models with gaps and a lot of distortion which makes it difficult to recognize the model as bad. We have called models which are complete but lack details and have malformed parts as good. Models with proper details and well-formed without any redrawn blocks are called excellent.

Test 2: Number of images taken and time to build

User Number	Number of Images	Depth Data captured	Time Taken (seconds)
1	5	Yes	90
2	15	No	100
3	60	Yes	130
4	102	Yes	200
5	20	Yes	100

Table 2: Time taken to build model with varying image count

This indicates that higher the number of images more time it takes for the model to be built. This makes sense because with more images the model has to identify points in more images and transform them to the model the object. Another interesting thing that we noticed was that different people clicked different number of images to build the model. The range is vast where user 1 clicked only 5 images whereas user 4 ended up clicking 102 images.

User feedback

We even asked users to provide us feedback about the app and these are the key insights we received from them:

1) Have a loading bar when stuck at loading screen so that the users have an idea of what’s happening
2) Add a loading screen at the deep linking part. Screen freezing for a second there
3) App skips a frame when you place the 3D object
4) Clicking images was tedious, recording video would be easier
5) Wait time for model building is too high

Insights from the above tests

From the above tests we can conclude that it is difficult to place and interact with a model if it has gaps. This makes sense because the model has empty space in the middle and does not respond when user taps on that spot. Both the users said model interaction is tough for this reason.

Another interesting observation from the above table is the time increase based on number of images. As the number of images increased the time taken to process and build a model also increases.

We can also conclude from the above table that depth data improves the model building capability but there is very limited data to support this claim.

Last observation that we can make is that in the case of a clean empty white background the object background is not included in the model unless the number of images are too low. This can be justified because if the number of images is too low like 5 in the above case then the small shadows in the white background is also considered as object but with increase in the number of images the background is removed from the model giving better results.

VII. RESULTS

Based on the above tests we tried some scenarios on our end and these are our observations.

A. Varying Images for Model Building

We ran various experiments to verify the results obtained from our system. We first tried building models with a varied number of photos starting from five and increasing by five in quantity. We observe from this experiment that as we increase the number of images the quality of models also improves. One more important observation is that we start getting well-formed models from 15 images. We went on and kept increasing the number of images to build the model and realized that with an increase in the number of images, the model kept getting better and more detailed. For our purposes having more than 15 good quality clear images covering the entire shoe is sufficient.

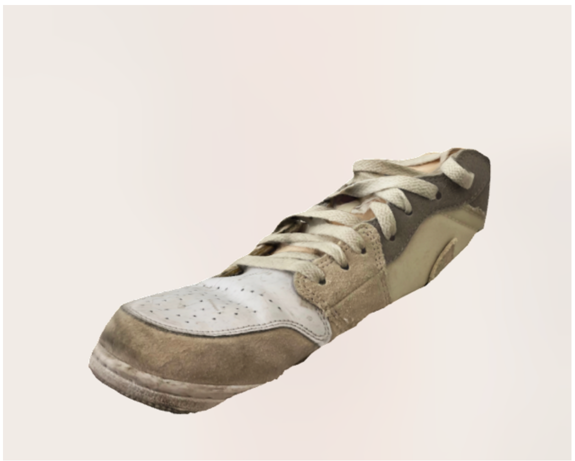


Figure 12: Model with 5 images and depth data

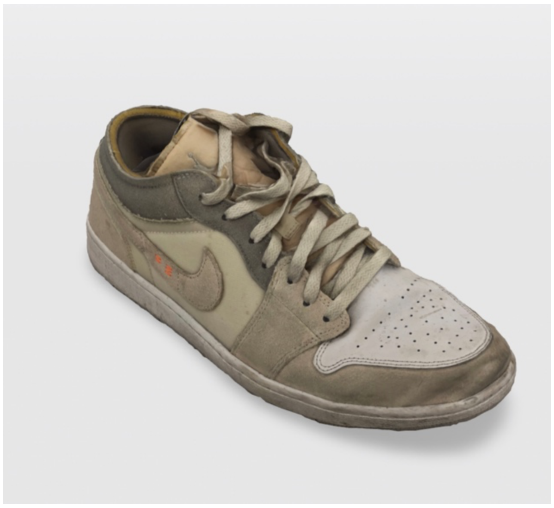


Figure 13: Model with 15 images and depth data

B. Model building with and without depth data

We ran another experiment on our project where we tried to build the model with and without image data. In our experiment, we observed that the model quality was better when depth data was included. This is in contrast to reading from the Grid Dynamics blog[6]. According to this blog the model quality did not get affected in case of clear images meaning that high quality images do not need depth data whereas, in the case of low quality images, the model building greatly improves with depth data. In our experiment, we found that the depth data is also useful in case we are working with a lesser number of images. We also read blogs and Apple forums where it is recommended to capture depth data.

C. Images with background disturbance

We ran another experiment where we tried to build the model with a normal background rather than a completely white background. This means that the images had some background disturbance which consisted of random objects and other stuff with different colors. The resultant model did not consist of the object but only the background objects. This implied that the images have to be free of disturbance otherwise it becomes difficult to recognize the object. We also tried building the model against a different colored and textured surface. This resulted in the entire surface being considered as the model. Hence we concluded that the model can only be formed using a completely white background free of disturbance otherwise the model will be skewed.

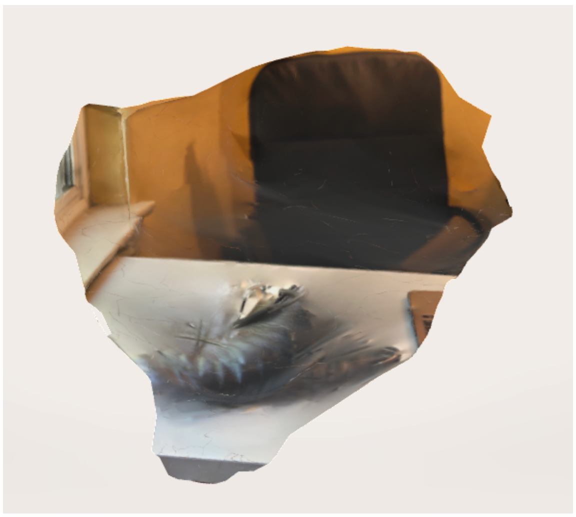


Figure 14: Model with background objects.

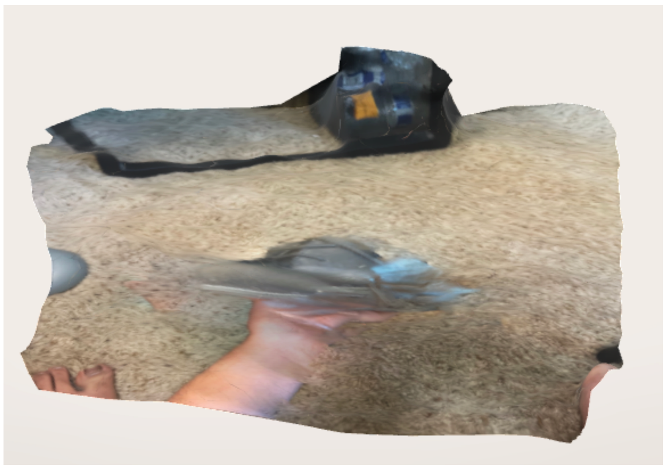


Figure 15: Hand-held object with textured background

VIII. TECHNICAL CHALLENGES

A. Establishing the connection between phone and server

The phone has to upload multiple images to the server for model building. The program for building a 3D model out of a 2D image is a native macOS app. So to set up a server we need a macOS system. This is available with Cloud providers at a higher cost. Hence we solved this problem by using the laptop as a server. The phone connects to the laptop which performs the model building operation. This is enabled by allowing the laptop to accept connections with any device on the network. Since this seems like a risky step we added verification before allowing users to connect and perform operations. We incorporate an object store like S3 in the middle to build to save the models for sharing between users.

Another issue with this is once the phone sends the image to the server the phone has to wait for the model to be returned. Model building is a slow process and hence to handle that better we establish a different connection for sending the model.

IX. CONCLUSION

3D AR Reconstruction is an app that combines model building and the AR user experience. We do not have existing solutions that let the users build models and also view them in the augmented reality environment. This project focused on implementing an app that would click images of the shoe and build a 3D model from those images and allow the user to view these shoes in augmented reality. 3D AR Reconstruction is an AR app that can essentially be useful for users. Augmented Reality is an upcoming and interesting field with a lot of scope in the future. It was challenging to establish the connection between the phone and the server for us. There exists solutions that let you experience premade models in augmented reality but nothing that lets you build the model.

This project required a lot of research and experimentation to get it up and running. This project's timeline can be broken into two ways for better understanding. We first broke down the project into different smaller components and tried building the most basic parts of it so we could understand the inner working and if there was a need for change in strategy. We worked on 2D image to 3D model building followed by sharing the model with different users and then explored ways to interact with this 3D model. All these parts done on simpler objects are the building blocks for our project next.

After building these smaller projects we decided to move on to building the final project. Since we had already figured out the smaller parts of the project and had overcome multiple roadblocks along the way we decided to start building our final app. We figured out the interactions between these components and how to transfer data between them. Since our project has a lot of intricate dependencies we came up with the server and phone design and the S3 connection between them. Doing so enabled us to allow model sharing using just links rather than sending the entire model via message or some other supporting medium.

REFERENCES

[1] Z. Oufqir, A. El Abderrahmani and K. Satori, "ARKit and ARCore in serve to augmented reality," *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2020, pp. 1-7, doi: 10.1109/ISCV49265.2020.9204243.

[2] L. Feng, X. Yang and S. Xiao, "MagicToon: A 2D-to-3D creative cartoon modeling system with mobile AR," *2017 IEEE Virtual Reality (VR)*, Los Angeles, CA, USA, 2017, pp. 195-204, doi: 10.1109/VR.2017.7892247.

[3] S. Mahurkar, "Integrating YOLO Object Detection with Augmented Reality for iOS Apps," *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2018, pp. 585-589, doi: 10.1109/UEMCON.2018.8796579.

[4] Y. Li, D. Hicks, W. S. Lages, S. Won Lee, A. Sharma and D. A. Bowman, "ARCritique: Supporting Remote Design Critique of Physical Artifacts through Collaborative Augmented Reality," *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, Lisbon, Portugal, 2021, pp. 585-586, doi: 10.1109/VRW52623.2021.00175.

[5] J. Zhou, Z. Xu, H. Yan, B. Gao, O. Yang and Z. Zhao, "AR Creator: A Mobile Application of Logic Education Based on AR," *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, Recife, Brazil, 2020, pp. 379-380, doi: 10.1109/ICVRV51359.2020.00109.

[6] xinReality (2016). "Photogrammetry". Available: https://xinreality.com/wiki/Photogrammetry. [Accessed: October 18, 2022]

[7] Ihor Mishchenko (2023). "Photogrammetry and augmented reality on mobile: A comparison of AR solutions" Available: https://blog.griddynamics.com/photogrammetry-and-augmented-reality-on-mobile/. [Accessed: April 11, 2023]

[8] Danny Moon (2017). "AR 101 — Learn the basics of Augmented Reality". Available: https://blog.viromedia.com/ar-101-learn-the-basics-of-augmented-reality-4b100e136242. [Accessed: September 20, 2022]

[9] Nghiaho(2020). "USING THE IPHONE TRUEDEPTH CAMERA AS A 3D SCANNER" Available: https://nghiaho.com/?p=2629. [Accessed: February 7, 2023]

[10] Yodayoda(2020). "From depth map to point cloud" Available: https://medium.com/yodayoda/from-depth-map-to-point-cloud-7473721d3f. [Accessed: February 7, 2023]

[11] Savannah Young(2023). "Inside the Growing Sneaker-Resale Market". Available: https://leaders.com/news/business/inside-the-growing-sneaker-resale-market. [Accessed: April 18,2023]

[12] Sarah Perez(2022). "Amazon gets into AR shopping with launch of 'Virtual Try-On for Shoes' ". Available: https://techcrunch.com/2022/06/09/amazon-gets-into-ar-shopping-with-launch-of-virtual-try-on-for-shoes. [Accessed: April 18, 2023

[13] Sheena Vasani(2023). "Snap launches a new business to help retailers with AR shopping". Available: https://www.theverge.com/2023/3/23/23651682/snap-enterprise-augmented-reality-artificial-intelligence-saas. [Accessed: April 25, 2023]