DEVELOPMENT OF AN APP FOR TRANSFORMING PHOTOS TO REALITY

Project Supervisor:
Dr. Pan Jia

Project Member:
Lee Jeng Heun (3035492642)
Wong Wing Lam (3035685029)

April 18, 2023
ABSTRACT

As the demand for 3D models in the applications associated with virtual reality has continuously increased, the limitations of traditional 3D modelling methods became very noticeable. The main problem with traditional method of constructing 3D models was that it required user to have firm understandings and experiences with the design tools, and invest considerable amount of time on the project. However, with the recently developed technique called Neural Radiance Field (NeRF), 3D object or scene could be generated from mere set of input photos. Development of NeRF certainly has reduced the burden on the user as the process itself has been simplified, however, to initiate and train a NeRF model still required knowledge on software engineering and deep neural networks. Therefore, the main objective of this project was to develop a user-friendly mobile application that anyone could use to turn photos to reality. In order to achieve this objective, various NeRF techniques were explored throughout the project. Ultimately, the main objective, mobile app to turn photos to reality, was successfully launched in a cloud server along with a feature to generate a Minecraft like environment for user to go into the reality released from their photos.
ACKNOWLEDGEMENT

I would like to express my gratitude to Quei-An, Chen (kwea123) for his Youtube tutorials and code release. Furthermore, I would like to express my deepest appreciation to the Nerfstduio team and the NeRF2Mesh team for sharing their wonderful improvision of NeRF. I would like to thank tamani-coding for sharing his tutorial on Three.js.

I would like to show my gratitude to my CAES teacher Locky Law for providing meaningful insights for writing this report. Last of all I would like to thank my groupmate Wong Wim Lam for his enthusiastic participation.
# TABLE OF CONTENT

ABSTRACT .............................................................................................................................. 1

ACKNOWLEDGEMENT .......................................................................................................... 2

LIST OF FIGURES ................................................................................................................ 5

LIST OF TABLES .................................................................................................................. 6

ABBREVIATIONS .................................................................................................................. 7

1. INTRODUCTION .................................................................................................................. 8
   1.1. BACKGROUND ............................................................................................................... 8
   1.2. SCOPE .......................................................................................................................... 8
   1.3. OUTLINE ..................................................................................................................... 9

2. LITERATURE REVIEW ....................................................................................................... 10
   2.1. PHOTOGRAMMETRY ................................................................................................. 10
   2.2. NEURAL RADIANCE FIELDS (NERF) ....................................................................... 10
       2.2.1. DIFFERENCE BETWEEN PHOTOGRAMMETRY AND NERF ......................... 11
   2.3. EXTENSION OF NERF .............................................................................................. 11
       2.3.1. NERF FOR SMOOTHER RESULT ...................................................................... 12
       2.3.2. 360 UNBOUNDED SCENES .............................................................................. 12
       2.3.3. ACCELERATE ..................................................................................................... 13
   2.4. NEURAL SURFACE RECONSTRUCTION .................................................................... 14

3. METHODOLOGY ................................................................................................................. 16
   3.1. WEB INTERFACE DEVELOPMENT ............................................................................ 16
   3.2. AI FOR TRANSFORMING IMAGES TO 3D MODEL .................................................... 17
       3.2.1. NeRF TRAINING ................................................................................................. 17
       3.2.2. MESH EXTRACTION .......................................................................................... 18
   3.3. VIRTUAL REALITY DEVELOPMENT ......................................................................... 18

4. PROGRESS, EXPERIMENTS, RESULT ............................................................................ 19
   4.1. TRY ORIGINAL NeRF MODEL ................................................................................... 19
       4.1.1. EXTRACTED MESH ............................................................................................ 19
   4.2. STUDY DIFFERENT NeRF METHODS ....................................................................... 21
4.2.1 EXPERIMENTAL RESULT ON SEVERAL NERF METHODS ................................................................. 21
4.3. WEB APPLICATION DEVELOPMENT .................................................................................................. 23
4.4. SET UP PRODUCTION ENVIRONMENT ............................................................................................... 24
4.5. IMPLEMENT WEBGL ............................................................................................................................ 25
4.6 IMPROVE MESH .................................................................................................................................... 26
4.7 INTEGRATION OF EVERY COMPONENT ............................................................................................... 27

5. CONCLUSION ......................................................................................................................................... 28
5.1 FURTHER DEVELOPMENT .................................................................................................................. 28

6. REFERENCES ......................................................................................................................................... 29
LIST OF FIGURES

Figure 1: Overview of NeRF 11
Figure 2: Rendering Method of NeRF and Mip-NeRF 12
Figure 3: (a) disparity of sampling point (b) contraction 13
Figure 4: Overview of instant-ngp 14
Figure 5: (a) coarse mesh extracted from trained NeRF. (b) fine mesh extracted after refining process. 15
Figure 6: System Dialogue of the Web Application 16
Figure 7: Overview of Web Interface 17
Figure 8: Workflow of NeRF model training 17
Figure 9: Workflow of Mesh Extraction 18
Figure 10: (a) colourless 3D mesh of FuTu Cow. (b) coloured 3D mesh after applying colouring code. 20
Figure 11: Workflow of user interface. (a) main webpage where user upload the image files. (b) new directory with the uploaded files created in the server. (c) code to run NeRF2Mesh using saved files. 23
Figure 12: Nerfstudio running in AWS EC2 24
Figure 13: overview of Web application environment 25
Figure 14: 3D scene built with Three.js 25
Figure 15: (a) Mesh extracted using NeRF_pl (b) Mesh extracted using NeRF2Mesh 26
LIST OF TABLES

Table 1: Performance of two method 21 – 22
ABBREVIATIONS

App: Application
AI: Artificial Intelligence
NeRF: Neural Radiance Fields
AWS: Amazon Web Server
EC2: Elastic Compute Cloud
1. INTRODUCTION

This chapter conveys an overview of the project. It first explains about the background of the project, followed by the problem definition and scope. Then it will wrap up with a brief outline of the paper.

1.1. Background

Before the development of computer graphics, photograph was the medium used by the people to cherish the moment of our life as it encapsulated our 3D world into a 2D image. But after the development of computer graphics, it is possible to display the 3D object in a digital space. Thus, instead of a 2D photo, one can capture the 3D reality into a digital world. This enabled people to travel the world in their room. The general method of obtaining a 3D object or scene is by using 3D graphics software such as blender or maya [2]. Using 3D graphics software to build fancy 3D models requires professional knowledge and a massive amount of time. Therefore, there has been studies on 3D reconstruction techniques like photogrammetry which generate 3D objects or scenes from photos [2]. Recently, researchers have proposed the method of NeRF which combined fully connected deep neural network and volume rendering to achieve state-of-the-art performance of view synthesis [3]. However, it is not easy for a user without proper software engineering knowledge to train a NeRF model and use it to reconstruct a 3D object or scene. Therefore, this project aims to explore the concepts and techniques of NeRF and build a mobile application embedded with NeRF technology. Ultimately, this project will help people to move from reality into a digitalized world without advanced knowledge on software engineering and deep learning.

1.2. Scope

The objective of this project is to develop a user-friendly app for generating a 3D object or scene by adapting the 3D reconstructing techniques. Using 3D reconstruction allow user to simply input a set of 2D photos to generate a 3D model. This app will allow user to ignore all the prerequisite knowledge one should have to execute 3D reconstruction methods.
In order to fulfil the above objective, this project is expected to deliver a mobile app with the following features at the end:

- Generate 3D models by inputting set of photos.
- A Minecraft like environment for user to go into the reality released from the photos.

1.3. Outline

The remainder of this report proceed as follows. Section 2 gives a detailed summary of the related paperwork on two different types of 3D reconstruction methods; Photogrammetry and Neural Radiance Fields (NeRF). Followed by a brief introduction of various extensions of NeRF and end with summary of Neural Surface Reconstruction techniques. Section 3 illustrates the steps taken to achieve the project deliverables. Section 4 reports about the progress and findings achieved throughout the project. Finally, section 5 wraps up the whole report with a conclusion and a future development at the end.
2. Literature Review

This section consists of the related literatures on the methods and concepts studied throughout this project. Here, two examples of 3D reconstruction method are introduced and compared. Then several extensions of NeRF is briefly explained, followed by detailed explanations of Neural Surface Reconstruction method.

2.1. Photogrammetry

Photogrammetry is simply a process of measuring an object by using multiple photographs [5]. But idea is stretched out to extreme when it is applied to creating a 3d model: Every single pixel in a photograph is measured in comparison to its neighbouring photos [5].

A Photogrammetry software generates 3D model under the following procedures: A set of images of an object from different angles are given as input [5]. Bunch of unique points called key points are generated in each of the images [5]. The software calculates the relative position of each image with its neighbours by comparing the key points [5]. After the position of photographs in 3D space is determined, a depth map is generated by comparing each image in detail with its respective pairs [5]. By using this depth map, the software calculates the actual location of the object in 3D space [5]. This derived object in 3D space is called mesh and it consists of millions of triangles [5]. After retrieving mesh, the software colours each triangle with their original colour of the image [5]. Finally, a 3D model with hundreds of millions of triangles are generated [5].

Some major limitation of Photogrammetry is that it requires very large resource and time to generate a 3D model. Also, it is difficult to process shiny or transparent objects due to reflection [6]. The result of Photogrammetry lacks consistency as its result depend heavily on how well the data was collected and processed [5].

2.2. Neural Radiance Fields (NeRF)

A NeRF model is generated from a specific viewpoint by following steps: firstly, for each pixel of an image, a ray is marched from its camera origin through the pixel and the scene to generate a sample set of 3D points [7]. Each sampling point will therefore have (x,y,z,theta, phi) which
are its position, ray origin and ray direction (see Figure 1 (a)). The sampling point will then go through positional encoding in order to represent high frequency function and pass into the 8-layer 256 neurons fully connected neural network to predict the colour and density of that sampling point (see Figure 1 (b)) [7]. Followed by using volume rendering techniques to assemble all sampling points of that ray using their colour and density to eventually calculate the colour of a pixel. And then optimize the model by comparing the predicted colour with ground truth colour of that pixel (see Figure 1 (c) to (d)) [7]. Result in a radiance field which input ray origin and direction then output the colour and density. However, that radiance field has no generalize ability which means that it can only apply for the scene used to train it.

Figure 1: Overview of NeRF [7]

2.2.1. Difference between Photogrammetry and NeRF

NeRF and Photogrammetry are very similar to each other as they both generate novel view of 3D object or scene from a set of 2D photos. However, the main difference between the two is that NeRF takes the positional and oriental data of the camera corresponding to each photo as input data as well. This allows NeRF to estimate the missing gap in-between each photograph with A.I. As a result, NeRF can generate images from new perspective that was not originally inputted whereas Photogrammetry cannot [6].

2.3. Extension of NeRF

After the publication of NeRF, there has been many studies and extensions that tackle the
limitation of the original Nerf such as long training time or poor performance in 360-degree scene. We studied the concept behind these extension in order to overcome the problems of NeRF and enhance our application’s performance. Below we go more depth into the extension that is related to our final deliverable.

### 2.3.1. NeRF for Smoother Result

One of the limitations of original NeRF is that when it trains images with poor resolution, the result of the training will be excessively blurred [10]. This is because NeRF renders the image by sampling the scene with a single ray per pixel (see figure 2 (a)) [10]. This means the colour of a pixel will be determined by the sampling point, which is the centre of that pixel, only. To resolve this problem Mip-Nerf uses anti-ialiased conical frustums (see figure 2 (b)) instead of single rays to render an image [10]. This method decides the colour of a pixel by relating it to its surrounding. By doing so, Mip-Nerf reduces the average error rate of training model by 17%, the training time by 7% and the size by 50% compared to NeRF [10].

![Figure 2: Rendering Method of NeRF and Mip-Nerf](image)

### 2.3.2. 360 Unbounded Scenes

For the original NeRF, it defines two planes which is near and far. The sampling point will generate along the ray equally within the two plane which means only the object in between the planes can be reconstructed. Therefore, NeRF++ proposed a method that is to generate constant number of sampling point and disparity the generation of sampling points along a ray such that there is more sampling point in closer distance [11]. Furthermore, mip-nerf360
maintains the concept of mip-nerf and use contraction to contract the frustums that are far away as there is no need to calculate such a large frustum [12] (see Figure 3).

![Image](image.png)

*Figure 3: (a) disparity of sampling point (b) contraction [8]*

### 2.3.3. Accelerate

The concept of hash encoding, and occupancy grid proposed in instant-ngp [9] greatly reduce the training time of NeRF. This method first use bounding boxes to wrap the scene (see Figure 4). For a sampling point, instead of positional encoding, instant-ngp uses hash function to retrieve the 8 features of the vertices of the voxel where that sampling point locates in for each bounding box. Then perform trilinear interpolation and input to the MLP. With the richer feature of a sampling point, the size of the neural network can shapely reduce to a concatenated 1-layer 64 neurons density MLP and 2-layer 64 neurons but colour MLP instead of an 8-layer of 256 neurons MLP while generating better prediction. Besides, as the sampling point will output sigma, the occupancy of a voxel can be calculated during the training process. For the voxel which are not occupied, the model will not generate sampling point in there which sharply reduce the sampling point that need to pass into the MLP [9].
Neural Surface Reconstruction is a widely used technique in computer vision and graphics. It operates by taking a set of input points or voxels and use neural network to reconstruct a 3D surface from them [22]. This technique is capable of reconstructing a detailed and accurate 3D scene from restricted or noisy set of inputs.

Neural Surface Reconstruction uses neural network and machine learning techniques like supervised learning or unsupervised learning to predict a continuous function. This function represents the points in a 3D space with regard to its corresponding positions on a surface [22]. After the training of neural networks is finished, the continuous function is the evaluated regard to the density of points [22]. This process makes sure that the detailed structure of the 3D object is captured. Finally, technique like mesh extraction is applied to generate the surface representation of the 3D object or scene [22].

Neural Surface Reconstruction has several advantages over the traditional surface reconstruction techniques in terms of accuracy and detail [22]. Also, it is capable of handling incomplete or noisy inputs. However, Neural Surface Reconstruction does have limitation as it requires large set of training data, takes long to train the neural network, and there is a chance of overfitting or underfitting of data [22].

NeRF2Mesh is an example of Neural Surface Reconstruction method, but it differs from other methods in terms of the way it represents 3D space and approaches the extraction of mesh [23]. With ordinary Neural Surface Reconstruction methods, the 3D space is discretised into a grid of voxels. The occupancy of each voxel represents its positional state with respect to the surface.
of the object. Whereas NeRF2Mesh uses NeRF representation to represent the 3D geometry of an object [23]. This enables NeRF2Mesh to accurately model complex objects with fine detail and texture. Also, NeRF2Mesh does not extract mesh directly from the learned function [23]. But instead, it extracts a low-resolution mesh called coarse mesh from the trained NeRF (see Figure 5(a)) and use it to initialise the refining process. During this process, coarse mesh is refined iteratively through joint optimisation of geometry and appearance. As a result, a high-resolution mesh called fine mesh is extracted [23] (see Figure 5(b)).

![Figure 5](image-url)
3. Methodology

A step-by-step approach of the project is presented here. The overview of this deliverable is to develop a web application to construct a 3D scene by using AI-empowered technology to transform photos into 3D model. To achieve this, Amazon Web Service Elastic Compute Cloud (AWS EC2) and Django is used as the infrastructure/framework to develop the web application, with NeRF2Mesh employed as its AI model (see figure 6).

![System Dialogue of the Web Application]

Figure 6: System Dialogue of the Web Application

3.1. Web Interface development

The initial step of the app development was to create a web application using Django framework. The user interface of this app takes set of images as an input and pre-process them into the right format (.png file) and size (below 720p). These images are then sent to the server for retrieving 3D model. Then the application was deployed in the cloud server to allow public access (see Figure 7). For this project, AWS EC2 was chosen as the platform of cloud server because it provides GPU instance with low cost and shows high reliability.
3.2. AI for transforming images to 3D model

After the server receives the images from the frontend, it is processed through two procedures - NeRF training and Mesh extraction. Once the procedures are completed, a 3D model is generated and saved in the server as an output.

3.2.1. NeRF Training

The first process of the AI is to train a NeRF model from the input images. To do this, it first generates the camera pose of each image by running it through the COLMAP, a commonly used pipeline for calculating camera pose for NeRF to handle custom dataset. The next step is to generate the dense depth of the image and enable the dense depth NeRF training. This is a useful technique when the number of input images are small because it provides NeRF more information about the real scene. Next, the NeRF starts training for several iterations to ultimately obtain a NeRF model used to render new view and Mesh extraction (see Figure 8).
3.2.2. Mesh Extraction

Once the Nerf model is completely trained, it can predict which voxel of the object is occupied in a 3D scene. NeRF2Mesh framework uses this NeRF model and further applies a marching cube algorithm to extract a coarse mesh (see Figure 9). Then the framework uses the coarse mesh as an initialiser to refine further through a deep neural network. As a result, a mesh with higher resolution called fine mesh is retrieved.

Figure 9: Workflow of Mesh Extraction [7]

3.3. Virtual Reality Development.

After successfully extracting the mesh and generating a 3D model from a set of photos, the first deliverable of the project has been completed. The next step is to create a Minecraft like environment for users to go into the reality released from the photos. To achieve this, the final stage of the project involved development of a virtual reality. The first step was to use WebGL to integrate 3D effect into the Django web app. Three.js was used to create a 3D scene in the web browser. This 3D scene was for loading 3D objects or scenes generated from photos and controlling the avatar to interact with the it in real time.
4. Progress, Experiments, Result

This section illustrates the progress and the development work conducted throughout the project, and the results achieve by the end of this project.

4.1. Try Original NeRF Model

At the beginning of the project, we spent most of the time studying the original version of NeRF through different explanation videos available on YouTube and Github. After understanding the theoretical part of NeRF, we moved on to train the original TensorFlow implementation of NeRF [9] using the dataset provided by the authors of NeRF in Google Colab. These datasets consisted of images and their corresponding camera poses. As a result, the trained model generated a .gif file, which illustrated new views of the scene (original: 20 images, gif: 120 images of different camera position).

After training with the provided data sets, we moved on to using our own sets of images to train NeRF model. During this phase, one of main challenge we encounter was calculating the camera pose of our dataset. To counter this challenge, we followed the advice presented in the code release of NeRF and tried to use COLMAP and LLFF to obtain the pose. However, the result was disappointing as there were constant errors when installing the COLMAP with its guideline. After some online searching and trials of different methods, we were able to solve this problem by installing the COLMAP through Anacoda. Furthermore, we used a Pytorch-lighting implementation of NeRF as it showed faster performance and more comprehensive explanations of its code [11]. As a result, a model was trained using the custom image set (FuTu Cow) in approximately 105 minutes and a .gif file was created by conducting inference on the trained model. (result [14]) The generated .gif file showed impressive performance. The new views were natural and smooth, which indicated that we can actually use the Pytorch-lighting implementation of NeRF as the AI model part of our deliverable.

4.1.1. Extracted Mesh

After using a set of FuTu Cow photos to train a NeRF model, we moved on to extracting the mesh from it. To achieve this, we applied the extraction code of the Pytorch-lighting
implementation of NeRF. As a result, a colourless 3D mesh of FuTu Cow was extracted (see Figure 10(a)). Additionally, we applied an original code written by the author of Pytorch-lighting implementation of NeRF designed to colour the colourless 3D mesh. The result showed reasonable similarity with the ground truth of the FuTu Cow that its shape was identifiable (see Figure 10(b)).

Figure 10: (a) colourless 3D mesh of FuTu Cow (b) coloured 3D mesh after applying colouring code
4.2. Study Different NeRF Methods

The original NeRF requires a huge amount of time in order to train a model and does not have generalise ability lead to a difficulty for using it to develop a application. However, we noticed that there were many extensions of NeRF to improve the result of nerf and tackle different of its limitation. For instance, the instant-ngp proposed by nvidia is famous for its ability to finish training a nerf model in 5 second. Therefore, we studied different nerf methods, some concept of those methods is stated in the literature review section.

4.2.1 Experimental Result on Several Nerf Methods

At the time of studying different nerf methods, we found nerfstudio which provides a simple API for training and testing various nerf method such as original nerf, mipnerf, instant-ngp, nerfacto. Nerfacto is a recommended method of nerfstudio for real world scene and it is continuously updated by the nerfstudio team. As far as we know, this method integrated the ideas of hash encoding from instant-ngp and contraction from mip-nerf360.

The experiment process originally conducted in Google Colab where GPU Tesla T4 was provided (see Table 1). But in order to use more free GPU resources, HKU GPU farm phase 2 with GPU RTX2080ti, cuda 11.2 was used. Phase 1 was not used because its GPU (RTX1080) didn’t meet the required compute capability for instant-ngp. Below is the experimental result on training model using instant-ngp and nerfacto for two dataset “poster” (provided by nerfstudio, a larger scene with 226 images in 135*240) and “cow” (our own photo of a doll with 11 images in 384*512). The evaluation matrix for the quality of image predicted is PSNR and SSIM while fps indicated the time for the trained model to inference images. The time in the table reports the time for training the model. The method are trained with the default parameter which can be view in the training log here [15].

<table>
<thead>
<tr>
<th>methods/dataset</th>
<th>poster</th>
<th>cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>instant-ngp</td>
<td>psnr: 16.77</td>
<td>psnr: 9.96</td>
</tr>
<tr>
<td></td>
<td>ssim: 0.50</td>
<td>ssim: 0.35</td>
</tr>
<tr>
<td></td>
<td>fps: 1.58</td>
<td>fps: 0.07</td>
</tr>
</tbody>
</table>
For nerf and mip-nerf, they took over 30 hours to train a model for each dataset. As using nerf needs huge amount of time to train a model while it does not have generalize ability, there is a training resource problem for using it to develop an application. However, the performance of instant-ngp and nerfacto both showed that with the method of acceleration, they can solve that problem. The performance of nerfacto outweighs instant-ngp especially in the larger scene (poster), although it needs a bit more time to train the model, its fps is much higher which is more favorable when our app wants to visualize the result of a nerf model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (min)</th>
<th>PSNR</th>
<th>SSIM</th>
<th>FPS</th>
<th>Time (min)</th>
<th>PSNR</th>
<th>SSIM</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>nerfacto</td>
<td>15</td>
<td>20.75</td>
<td>0.88</td>
<td>14.26</td>
<td>24</td>
<td>11.61</td>
<td>0.37</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 1: performance of two methods*
4.3. Web Application Development

For this project, Django web frame was used to develop the user interface for uploading a set of image files used as input data to train the 3D object or scene. When the user uploads the files to the web application (see figure 11 (a)), the Django app automatically create a new directory in the server and saves the files there (see figure 11 (b)). Then this directory is used by the subprocess code to launch NeRF2Mesh training (see figure 11 (c)).

Figure 11: Workflow of user interface. (a) main webpage where user upload the image files. (b) new directory with the uploaded files created in the server. (c) code to run NeRF2Mesh using saved files.
4.4. Set up Production Environment

We have set up the AWS EC2 so that our app would allow public access and be able to run the NeRF code to train the model on the cloud server. The instance we used is g4dn.xlarge in US-East which has 1 Tesla T4 GPU and 16Gb ram with the price USD 0.52 per hour on demand of use [15]. This instance type was chosen because it was one of the least expensive instance types compared to other options. Also, its GPU met the capability requirement of 7.5 to train instant-ngp. Additionally, according to our experience, instant-ngp works well in Google Colab which is also using Tesla T4 as the GPU and the same amount of RAM.

Compared to the HKU GPU farm and Google Colab, AWS EC2 as an IaaS does not have cuda preinstalled, which is necessary for using GPU in order to train NeRF. Thus, we had to install it from scratch. Eventually, after trying several methods, we were able to install cuda by installing nvidia driver first and then install cuda 11.4. from it. After that, we installed the python package of nerfstudio and successfully ran it in the AWS server (see Figure 12). This meant that the environment to train deep learning model was ready.

![Figure 12: Nerfstudio running in AWS EC2](image-url)
Finally, we installed the Django package and deployed the Django project into EC2. After opening up the 8000 port, we were able to access the web app through the IP of the EC2 instance (see Figure 13).

![Web server diagram](image)

*Figure 13: overview of Web application environment*

### 4.5. Implement WebGL

In addition to building a web app that allows users to upload their images and obtain 3D models without thinking the technique behind, we also aimed to create a virtual reality environment where real-world 3D models can be visualized and interacted with. To achieve this, we explored the WebGL technique, which provided web pages with robust graphical effects. We used Three.js to construct a 3D scene, load 3D objects, and control character movement. However, integrating the Three.js library into the Django framework was challenging because we couldn't import the Three.js as a module of a JavaScript file without a Node.js environment. Eventually, we found a way to do this by using webpack to pack the JavaScript file with its library into a single JavaScript file that could be used like a normal JavaScript file. After that, we used Three.js to create a 3D scene that could load 3D models and control character movement (see Figure 14).
4.6 Improve Mesh

NeRF give good performance on view synthesis, however, the performance on surface reconstruction is relatively weak (see Figure 15(a)). To have more interaction, a 3d model would be a better choice than volume rendering images. Thus, we tried to find how to improve the performance of mesh. We found that there are several neural surface reconstruction methods in sdfstudio which is built on top of nerfstudio, which means they can run in the same environment we set up before. Those methods improve the surface reconstruction performance on NeRF by applying SDF into it. At the time when we tried to put the colour into the mesh generated by NeuS, we found a recently proposed methods called Nerf2Mesh which can achieve an even better surface reconstruction performance and can calculate the texture of the mesh (see Figure 15(b)). Therefore, we used NeRF2Mesh as the AI model of the final deliverable.

Figure 14: 3D scene built with Three.js
4.7 Integration of Every Component

The final work of the deliverable is to integrate each components mentioned above including a simple web interface, AI, virtual reality into a complete web app. By doing this, we built a web app that generate 3D models by inputting set of photos. A Minecraft like environment that user can put the 3D model into it, visualize and download them from there.
5. Conclusion

This project developed a user-friendly mobile app which transform photos to reality.

During the first semester, we focussed on studying a 3D reconstructing technique called NeRF. We studied the technique of NeRF by reading literatures and trying the code release of NeRF methods. Also, we successfully trained NeRF models to render new views and obtain 3D object using both provided data set and our own photos. During this phase, we found several novel extensions of NeRF which improved the performance of it. We spent a lot of time looking at extensions of NeRF and deepen our understanding about the concept of NeRF methods.

In the second semester, we moved on to developing a mobile app which allow users to transform their photos into reality. We used Django to build a web app and AWS for the cloud server. NeRF2Mesh method was applied as the AI technique used to transfer image files to 3D object or scene. Moreover, ThreeJS was used to create web based virtual reality that can put the real-world object into the digital world.

In summary, this project developed a 3D reconstruction virtual reality application in Django framework for user to go into the reality release from the photos base on the technique of NeRF and WebGL and was deployed into AWS.

5.1 Further Development

In the future, the team will study on the volume rendering technique of ThreeJS/Unity so that the virtual reality will have better visual performance on displaying the 3D scene or object generated from NeRF. Also, we will try to develop the virtual reality part such that the user can have more interaction with the 3D scene or object of the virtual reality. Furthermore, we will continue studying the extension of NeRF so that the app can have more effect such as reconstructing dynamic scene.
6. References


