DEVELOPMENT OF AN APP FOR TRANSFORMING PHOTOS TO REALITY

Project Supervisor:
Dr. Pan Jia

Project Member:
Lee Jeng Heun (3035492642)
Wong Wing Lam (3025685029)

January 22, 2023
ABSTRACT

With the spread of Virtual Reality (VR) games, people have become accustomed to the concept of 3D modelling. As a result, many tools for generating 3d models were introduced to the market. However, after experimenting with many of these tools, we learned that most of them were very complicated and required a sufficient amount of studying to master. This project aims to learn the state-of-art view synthesizing technique -Neural Radiance Fields and use this technique to develop a user-friendly app which will generate a 3D model/scene from a set of 2D images. This will save the users from spending excessive time and money on creating 3D scene in the digital world. The project will be divided into two main phases – learn NeRF methods and implementing it to the mobile app. For the time being, we have successfully trained different NeRF model from the source code with our own set of data. And we will be working on the viewer which use a trained model to do inference and deploy a Django app combined with NeRF into the cloud server.
ACKNOWLEDGEMENT

We would like to express our gratitude to Quei-An, Chen (kwea123) for his Youtube tutorial video and code release. We would also like to express our gratitude to our project supervisor, Dr. Pan Jia, for his guidance.
# 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. **PROBLEM DEFINITION** .................................................................................................................. 8

    1.3. **SCOPE** ............................................................................................................................................ 8

    1.4. **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

3. **METHODOLOGY** ...................................................................................................................................... 14

    3.1. **NeRF MODEL TRAINING** ............................................................................................................... 14

    3.2. **MESH EXTRACTION** ...................................................................................................................... 14

    3.3. **MOBILE APP DEVELOPMENT** ....................................................................................................... 15
4. PROGRESS ........................................................................................................16

4.1. TRY ORIGINAL NeRF MODEL ........................................................................16
4.2. STUDY DIFFERENT NeRF METHODS ..............................................................16
4.3. EXPERIMENTAL RESULT ON SEVERAL NeRF METHODS ..............................16
4.4. CREATE AN UNREAL SCENE ........................................................................18
4.5. START DEVELOPING THE MOBILE APP ......................................................18

5. ON-GOING WORKS ..........................................................................................19

6. FUTURE PLAN ..................................................................................................20

7. CONCLUSION ....................................................................................................21

8. REFERENCES .....................................................................................................22
# LIST OF FIGURES

- Figure 1: Overview of NeRF 11
- Figure 2: Rendering Method of NeRF and Mip-NeRF 12
- Figure 3: spacity of sampling point, contraction 12
- Figure 4: overview of instant-ngp 13
- Figure 5: Workflow of NeRF model training 14
- Figure 6: System Dialogue of the Mobile App 15
- Figure 7: style transfer result 18
- Figure 8: Django app for uploading and storing video file 18
LIST OF TABLES

Table 1: Performance of two method

Table 2: Project Schedule
ABBREVIATIONS

App: Application
AI: Artificial Intelligence
NeRF: Neural Radiance Fields
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

The concept of 3D modelling has experienced a significant development over the last few decades and manage to successfully wrap around our everyday lives (i.e. VR games, HoloLens, 3D printing). As a result, many 3D modelling tools and 3D reconstruction techniques (i.e. NeRF, Photogrammetry) were introduced in the market.

3D modelling is the process of reproducing a three-dimensional representation (3D model) of an object with a specialized software [1]. These 3D models can be generated from either existing objects or imaginations of the designer [1]. 3D models are used in variety of industries including VR, gaming, marketing, architectural, 3D printing and medical imaging [2].

3D reconstruction is the method of reconstructing an object within a virtual 3D space with a computer [3]. There are many ways to conduct 3D reconstruction but the most common method is to take a set of 2D photos as input data or scan the actual object [3].

1.2. Problem Definition

Currently, there are a large variety of tools to generate 3D models in the market. However, these tools have severe issues regarding cost efficiency (several thousands of pounds per user per year) as well as time and resource consumption (requires team of specialists with proper knowledge) [4].

1.3. Scope

The objective of this project is to develop a user-friendly app for generating a 3D model by adapting the 3D reconstructing techniques. Using 3D reconstruction will allow users to simply
input a set of 2D photos to generate a 3D model. This will save both time and money users would have wasted on learning the skill required to run the existing tools.

In order to fulfil the objective, we plan to deliver the following features at the end of our project: Implementation of 3D reconstruction on the app to allow users to generate their own 3D models; A mobile app where users will input photos to be sent to the server; A web server where the actual 3D reconstruction will happen; A system between the web server and the app for instantaneous communication.

1.4. 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 brief introductions of various extensions of NeRF. Section 3 illustrates the methodologies used to explain the steps taken to achieve the project deliverables. Section 4 reports on the project status including the ongoing work on implementing NeRF on the app and future plans on the app development. Finally, section 5 wraps up the whole report with a conclusion.
2. Literature Review

This section consists of the related literatures on the technologies studied throughout this project. It highlights the concept of one example of 3D reconstruction called Photogrammetry and the limitations associated with it. It then explores another example of 3D reconstruction called NeRF in detail and how it overcomes the limitations encountered by Photogrammetry.

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]](image)

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. Here we discuss some of the extensions.
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-aliased 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% compare 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].

![Figure 3: (a): sparsity of sampling point (b) contraction](image)
2.3.3. Accelerate

The concept of hash encoding and occupancy grid proposed in instant-ngp [] greatly reduce the training time of NeRF. This method first use bounding boxes to wrap the scene. 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. [7]

![Figure 4 overview of instant-ngp [7]](image)
3. Methodology

A step-by-step approach of the project will be presented here. It consists of two main phases – implementation of NeRF and deploying the actual app. Throughout the project, python will be used as the primary programming language because we are familiar with developing app with python from our experience. Also, python has a machine learning library called pytorch which is generally used in the source code of the machine learning model we are using.

3.1. NeRF Model Training

During the phase one of the project, we will follow the source code provided by the actual developers of NeRF and Google Colab to train our own NeRF model. A NeRF model is built under the following procedures: first, it takes a set of 2D photos to train and continuously run these sets throughout the training process while optimizing the loss of radiance generated at each pose (see Figure 5). Since NeRF requires the positional and the oriental data of the camera corresponding to the photo, we will run the initial process with the input photos. Then we will follow the procedures listed in the NeRF repo and use COLMAP to generate the pose of images [8].

Once the positional and orientational data of the photos are generated, they will be used to train the NeRF model. The result of this train model will be a radiance field which predict the colour of a ray at any direction such that a new novel view of the input photos will be generated.

![Figure 5: Workflow of NeRF model training [7]](image)

3.2. Mesh Extraction

In order to extract a 3D mesh, our program need to have the knowledge on the locations occupied by the object first. To do this, it will generate a cuboid-like grid volume to cover the entire object. Once the grid volume is generated, NeRF model will be used to guess the
occupied state of each cell. Then the marching cube algorithm will be used to extract the 3D mesh. The extracted mesh consists of polygons (vertices and faces) only and lack the information about the colour of the object. Before retrieving the colour of mesh, the program needs to discard any access noise that may have been generated due to incorrect calculation of occupancy in the beginning. This is done by simply keeping the largest group of polygons that are connected. Once the noise has been removed, the RGB values of the mesh will be retrieved by projecting it to the NeRF model. When the 3D mesh is projected onto the NeRF model, the software will ignore the faces of the polygons and consider the vertices only. This is because the RGB value of a vertex is the average of the two neighbouring faces. Once the RGB values have been assigned to each vertex, they will be averaged in order to retrieve the final colour of the 3D mesh. [9]

3.3. Mobile App Development

During the Phase 2 of the project, a Django application will be developed where users may use to upload the videos or images that they want to generate the 3D model. Then Linux web server will be created where the model training will be conducted. Linux web server is used because it is easier to install the NeRF library. Once the model is trained in the backend server, a 3d mesh file will be generate and provided to the user for download. The users will be provided with the options of using the app’s visualiser installed on the web page to display the 3D mesh file or just download the file and use their own visualiser. As for view synthesis, the backend server can render the image in particular viewpoint base on a trained model and use our frontend interface as a viewer to view it. (see figure 6)

![System Dialogue of the Mobile App](image-url)
4. Progress

This section illustrates the progress we have made so far throughout the project.

4.1. Try Original NeRF Model

At the beginning of the project, we studied the original version of NeRF through different explanation videos available on YouTube which we encultured deeper into the principle of NeRF. Furthermore, we trained a Pytorch-lightning implementation of NeRF [9] from the dataset provided by the nerf authors which includes images and their poses. And we were successful in creating a view synthesis video using that trained model.

After creating the view synthesis with the sample data, we moved on to using our own sets of images to train NeRF model and this process required to create pose for our own images. By that time, we tried to follow the advise from the code release of nerf to use COLMAP and LLFF to obtain the pose of our images. However, in that stage of development, we consistently received an error on installing and executing COLMAP following the installation guide of COLMAP[11]. This problem was eventually solved by installing COLMAP through Anacoda. As a result, a model using our set of images was trained in approximately 105 minutes and a .gif file was created by using the trained model to do inference. (result [14])

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.

4.3. 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. 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>
<tr>
<td></td>
<td>time: 15 min</td>
<td>time: 13 min</td>
</tr>
<tr>
<td>nerfacto</td>
<td>psnr: 20.75</td>
<td>psnr: 11.61</td>
</tr>
<tr>
<td></td>
<td>ssim: 0.88</td>
<td>ssim: 0.37</td>
</tr>
<tr>
<td></td>
<td>fps: 14.26</td>
<td>fps: 0.46</td>
</tr>
<tr>
<td></td>
<td>Time: 24 min</td>
<td>Time: 23 min</td>
</tr>
</tbody>
</table>

*Table 1: performance of two methods*

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.
4.4. Create an Unreal Scene

During our study on NeRF, we thought about some features that could be included in the app that could entertain the users. Then we came across an interesting technique called Artistic Style Transfer GAN. The exciting function of this technique is that it creates cartoon like images (see Figure 7 (c)) by merging a realistic image (see Figure 7 (a)) with an artistic style image (see Figure 7 (b)). By including this function in our app, it may able to generate not only realistic views but also views with various textures because the pose of the images is kept unchanged.

![Figure 7: a) original image, b) style image to apply, c) result of applying style transfered image and visualize in the viewer of nerfstudio](image)

4.5. Start Developing the Mobile App

We have developed a django app that allows user to upload video file to the local server (see Figure 8 (a)) and store it in the backend database (see figure 8 (b)). Also, we created an AWS account and went through its tutorial on deploying the django project to the AWS Elastic Beanstalk to get ready for our actual app.

![Upload Video to Generate 3D Mesh](image)

![Figure 8: Django app for uploading and storing video file](image)
5. On-Going Works

We are setting up cuda environment for the AWS GPU instance so that our app can run the NeRF code and training the model on the cloud once the users upload their images or video. The instance we are using is g4dn.xlarge in US-East which has 1 Tesla T4 GPU and 16Gb ram in the price USD 0.52 per hour on demand of use [15]. We chose this instance type in US-East because it is one of the least expensive instance types compared to different GPU instances provided in other regions. Also, its GPU has enough compute capability to train nerfacto and instant-ngp according to our experience in Google Colab which uses the same GPU.
6. Future Plan

In order to properly deliver project scope over time, the project is divided into multiple phases along with its goals and internal deadlines (see Table 2). Since we have successfully trained the model, we are moving on to using it to generate an inference. We plan to finish creating a viewer on the Django app by the early February. This viewer will be used to render image and 3D mesh and visualised them in the web page. Then, we will combine nerf training and inferencing with our Django app and build the UI for the mobile app. If we finish all these tasks before April, we will use the rest of the time to find a way to combine nerf model with unity in order to create mixed reality application like this demo [9]. Moreover, we will continue studying the extension of NeRF such as SNERG for real time rendering [16] and Block-NeRF for very large-scale environment [17].

<table>
<thead>
<tr>
<th>Time</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 Oct 2022</td>
<td>project planning (done)</td>
</tr>
<tr>
<td>08 Jan 2023</td>
<td>Deliverables of Phase 1 (Main)</td>
</tr>
<tr>
<td></td>
<td>● Study the concept of NeRF and start implementing it (done)</td>
</tr>
<tr>
<td>22 Jan 2023</td>
<td>interim report (done)</td>
</tr>
<tr>
<td>15 Feb 2023</td>
<td>Deliverables of Phase 2 (Main)</td>
</tr>
<tr>
<td></td>
<td>● Finish deploying the backend server (pending)</td>
</tr>
<tr>
<td></td>
<td>● Start building frontend material (pending)</td>
</tr>
<tr>
<td>01 April 2023</td>
<td>Deliverables of Phase 3 (Advanced)</td>
</tr>
<tr>
<td></td>
<td>● Combine Mixed Reality with the app</td>
</tr>
<tr>
<td>18 Apr 2022</td>
<td>Final report</td>
</tr>
</tbody>
</table>

*Table 2: Project Schedule*
7. Conclusion

This final year project aims to learn the technique of NeRF and develop a user-friendly mobile app to transform 2D images into a 3D scene/model. During the first semester, we looked at different types of 3D reconstructing techniques. Among those techniques we have chosen to focus on a technique called Neural Radiance Field because it showed highest potential. We started by trying the original NeRF method and train a NeRF model using our own data. During this session, we noticed that there are a lot of novel extensions of NeRF which improve the performance of NeRF. We spent a lot of time looking at extensions of NeRF to deepen our understanding about the concept of NeRF methods which is our primary objective.

Once we successfully trained our own model using advance methods of NeRF, we moved on to developing a Django mobile app which allow users to transform their photos into reality. We are now working on a method to train the NeRF model in our cloud server and develop a viewer that will be used to visualise the result back to the user. We will learn more nerf extension particularly for first rendering and large-scale environments. Also, integral the nerf model into unity for developing mixed reality. By the end of this project, we believe we will be able to develop our own user-friendly 3D reconstruction application.
8. References


