3D-object reconstruction methods

A comparative study

1. Introduction

The reconstruction of 3D-objects from images has been a vibrant area of research for few decades. Initially, the process involved the use of specialized hardware and software, such as laser scanners, to create 3D-models. Over time, the focus has shifted to more computationally based methods. One popular technique is Photogrammetry, which leverages images to extract information about the 3D-structure of an object. While Photogrammetry has proven to be an effective method for 3D-object reconstruction, it can be time-consuming. In our previous assignment, we utilized Meshroom, which doesn't make use of GPUs to great extent. However, new software such as 3D-Zephyr has customized their methods to take advantage of such performance-enhancing GPUs, and it is one of the tools that we will explore for our use case.

While Photogrammetry tools can produce great results, they do have limitations such as the requirement for high-quality images, issues with occlusions, shadows, and reflections, and potential errors in image matching and processing. However, with the advancements in computer vision and deep learning, a variety of new techniques have emerged, such as Neural Radiance Fields (NeRFs). The advent of NeRFs has revolutionized this field, allowing for the creation of 3D-environments from 2D-images within seconds or minutes. While NeRFs are not yet commonly used to create 3D-objects, various algorithms have been developed to convert NeRF outputs into 3D-meshes. This technique can be a game-changer if it performs well, providing an efficient and practical means of generating 3D-objects from images. In this project we will explore three such methodologies and compare their results with photogrammetry output.

2. Theoretical background

The process of reconstructing 3D-objects from 2D-images has been an active area of research for several decades. Initially, the methods involved the use of specialized hardware and software, such as laser scanners and structured light systems, to create 3D-models. Fig1 shows one such example of a 3D-scanner. Back in 1998 they were called as the portable scanners which ironically used to weigh around 40-50 kgs.

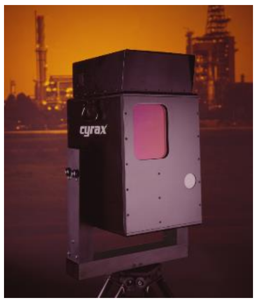


Fig1. Cyrax Scanner (**source**)

In Fig2 below we can see the results from such scanners. These results were highly accurate and to this day are quite good and at par with new scanners.

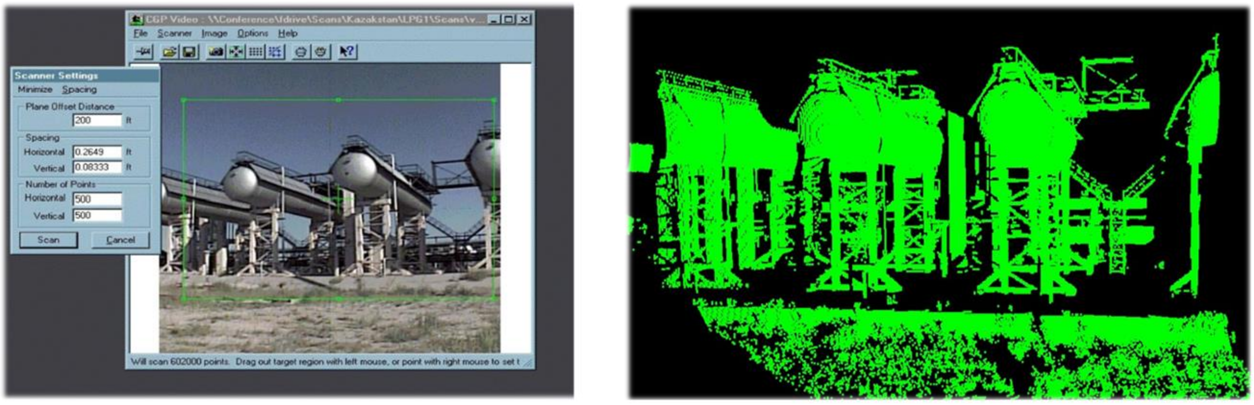

CGP Video : \\Conference\fdrive\Scans\Kazakstan\LPG1\Scans\v...
File Scanner Image Options Help
Scanner Settings
Minimize Spacing
Plane Offset Distance 200 ft
Spacing
Horizontal 0.2649 ft
Vertical 0.08333 ft
Number of Points
Horizontal 500
Vertical 500
Scan Cancel
Will scan 602000 points. Drag out target region with left mouse, or point with right mouse to set t

Fig2. Results from Cyrax Scanner(**source**)

3D-scanners are still used and have improved by leaps and bounds like every other technology. From Cyrax scanner being 51 kgs to latest scanner from Leica - BLK360, which is barely 1kg. The scan speed has also improved from 2000 points per/sec to 360,000.[1]

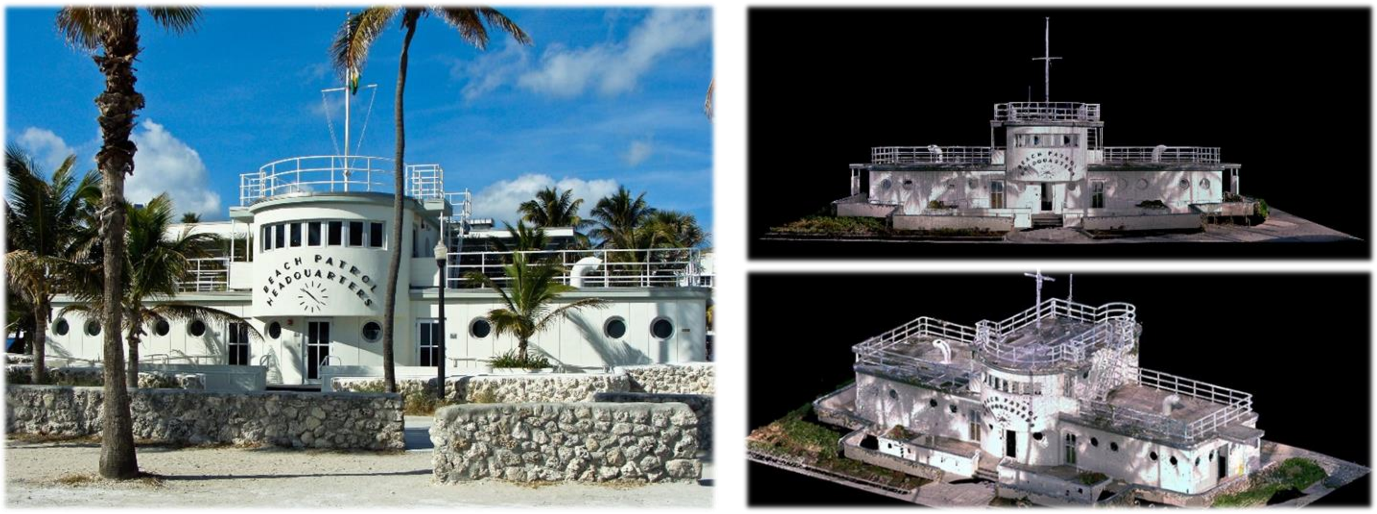


Fig3. Here are results from Leica – BLK360(**source**)

But not everyone can afford these devices and they are quite a lot expensive. But with advancement of computer vision and deep learning, a variety of new techniques emerged, making 3D-object reconstruction more accessible and practical. One of the most popular techniques is **Photogrammetry**, which involves the use of images to extract information about the 3D-structure of an object. Photogrammetry has been used for some time now and has proved to be an effective method for 3D-object reconstruction.

The process of photogrammetry involves taking multiple images of an object from different angles and using the information in these images to reconstruct a 3D-model of the object. This is done by identifying common features in the images, such as corners or edges, and using these features to triangulate the position of the object in 3D-space. Once the position of the object is known, the shape of the object can be reconstructed using techniques such as stereo reconstruction, structure from motion, or multi-view stereo.

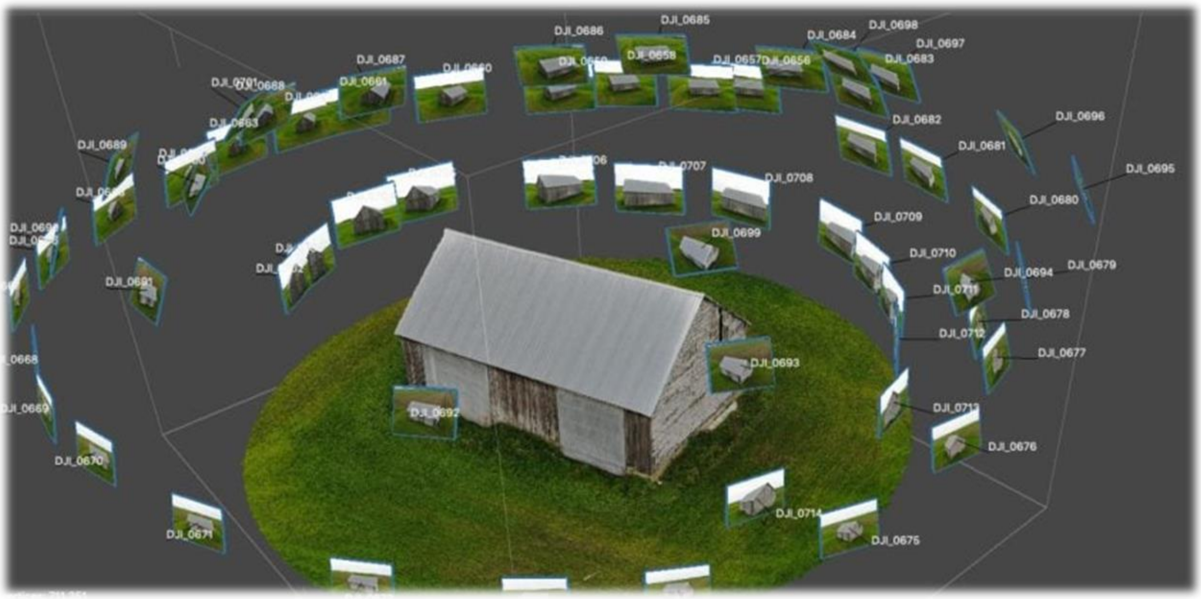


Fig4. A photogrammetry software being used to constructing a 3D-mesh object using 2D-images (**source**)

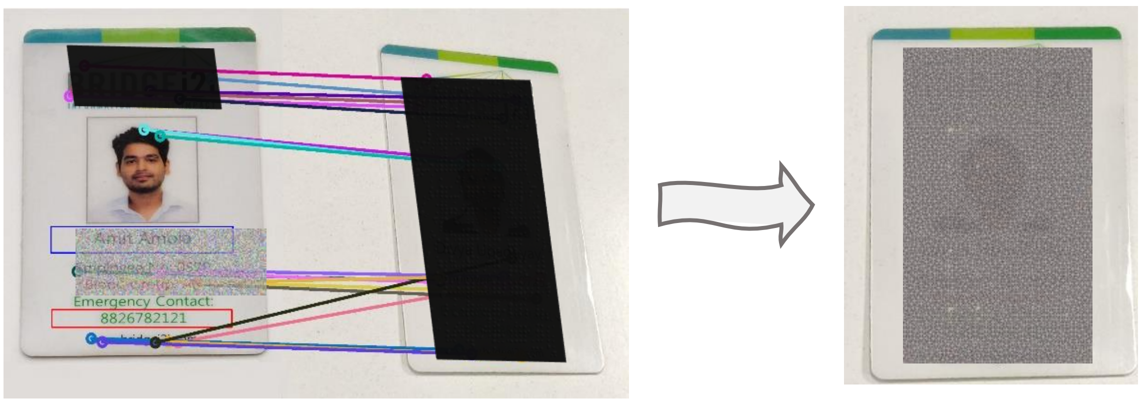
Let's take an overview of how Photogrammetry works:

Photogrammetry

Photogrammetry is the process of creating 3D-models or maps from 2D-images. The photogrammetry pipeline usually involves several steps, including image acquisition, camera calibration, feature detection, image matching, 3D-reconstruction, and mesh generation. Let's understand them briefly:

- **Image Acquisition:** The first step in a usual photogrammetry pipeline is to acquire a set of images of the object or scene. The images can be captured using a variety of imaging devices, such as digital cameras, smartphones, drones, or satellites. To obtain high-quality results, it is essential that the captured images are overlapping, taken from different angles and viewpoints.
- **Camera Calibration:** The second step is to calibrate the camera or imaging device which involves determining the intrinsic and extrinsic parameters of the camera, such as focal length, distortion coefficients, and camera position and orientation. Calibration is essential to correct lens distortions, adjust camera settings, and ensure accurate measurements and 3D-reconstructions.
- **Feature Detection:** The third step is to detect feature points in the images. Feature points are distinctive regions in the images that can be easily tracked and matched between different images. Feature detection algorithms, such as SIFT, SURF, or ORB, can automatically perform this task and extract their descriptors.

  We can in fact do all kinds of stuff with these detected features. One interesting use case is of Homography based image translation. Here's an example below where features are matched in two images and then the second image is translated in same orientation as the first one. And this is exactly the next step.



- **Image Matching:** The fourth step is to match the feature points between different images. Image matching involves finding corresponding feature points in different images and estimating their relative positions and orientations. Matching algorithms, such as RANSAC, can robustly estimate the camera pose and reject outlier matches.

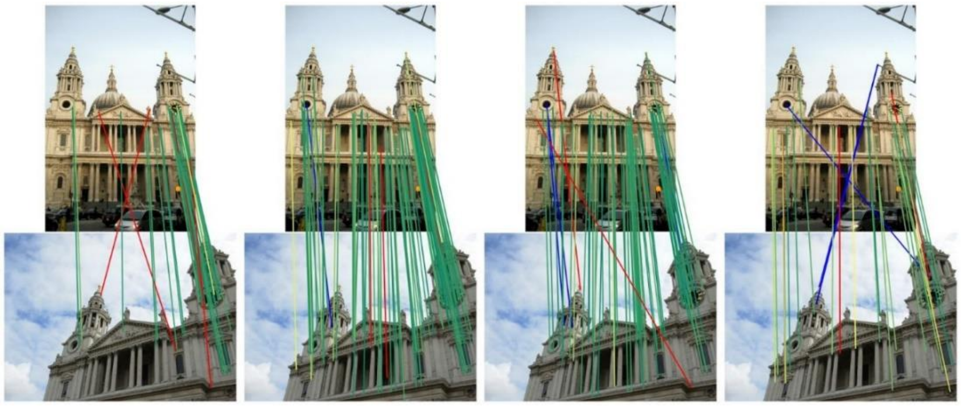


Fig5. An example of how for different algorithms are able to match features in the building's two images from different angles(**source**)

- **3D-Reconstruction:** The fifth step is to reconstruct the 3D-structure of the object or scene from the matched feature points. 3D-reconstruction involves triangulating the matched feature points to estimate their 3D-positions in space. Reconstruction algorithms, such as structure from motion (SfM) or multi-view stereo (MVS), can estimate the 3D-structure of the scene from multiple images.[2]

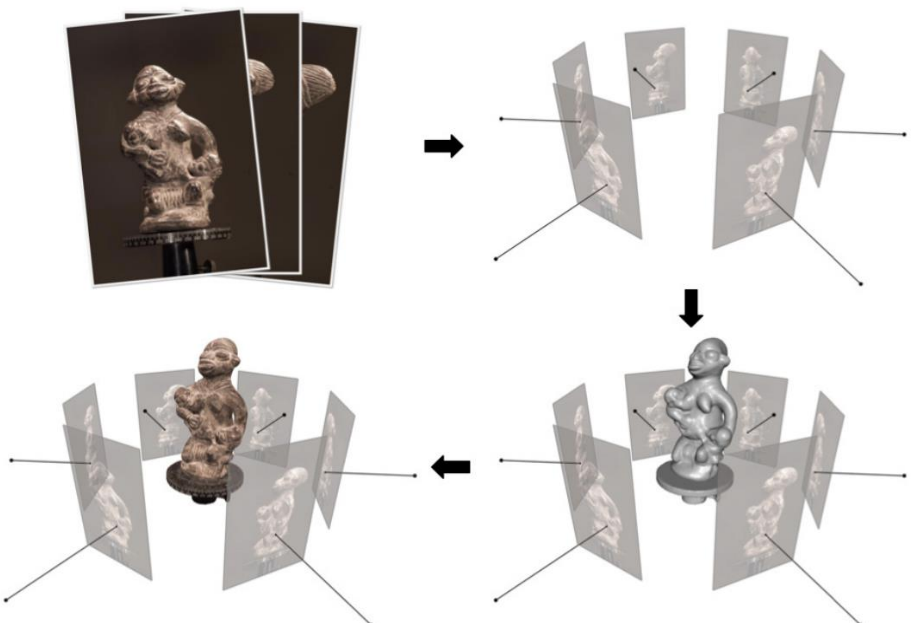


Fig6. Example of a multi-view stereo pipeline. Clockwise: input imagery, posed imagery, reconstructed 3D-geometry, textured 3D-geometry(**source**)

- **Mesh Generation:** The final step is to generate a mesh model from the 3D-point cloud. Mesh generation involves creating a surface representation of the 3D-point cloud by connecting the neighbouring points with triangles or other polygonal shapes. Mesh generation algorithms, such as Poisson surface reconstruction or marching cubes, can generate smooth and watertight mesh models from the 3D-point cloud.

In last decade we have seen several such software and tools such as Meshroom, Reality Capture, 3D-Zephyr, Metascape, etc.

Advances in technology have made such software more accessible, particularly with the advent of smartphones with built-in LiDAR capabilities. With high computation and depth information available, it has become easier to create 3D-objects using only the device in your hand. This is nothing but using photogrammetry in a mobile application and as with other photogrammetry tools, this approach has similar limitations as well.

Additionally, not everyone can buy an expensive device with such technology either. One can take care of the lighting and condition around which the photographs are taken and avoid any object which is shiny or has reflections. But that indeed is not always possible either. So, what is the solution? – enters **NeRFs**.

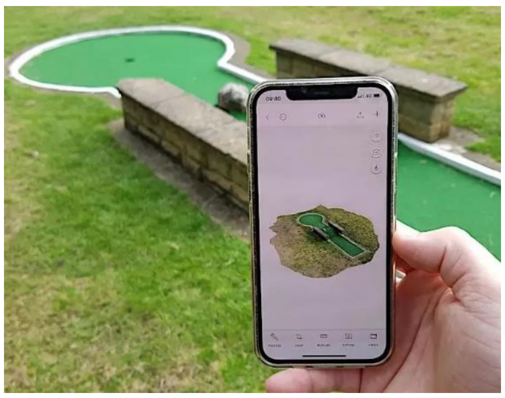


Fig7. An iPhone application Polycam being used to create 3D-model(**source**)

In recent years, the advent of **NeRFs** has revolutionized the field of 3D-reconstruction. NeRFs are a type of deep learning algorithm that can be used to create 3D-environments from 2D-images within minutes. Unlike photogrammetry, NeRFs do not require multiple overlapping images from different angles. Instead, they can reconstruct objects from few images captured from different viewpoints. Let's have a brief overview of NeRF's process:

Neural Radiance Fields (NeRFs)

NeRF is a recently developed technique for 3D-modeling and rendering of real-world objects and scenes from 2D-images. It uses a neural network to learn the 3D-structure and appearance of an object or scene directly from a set of 2D-images. Here's an overview of the pipeline for creating 3D-object mesh using NeRF:

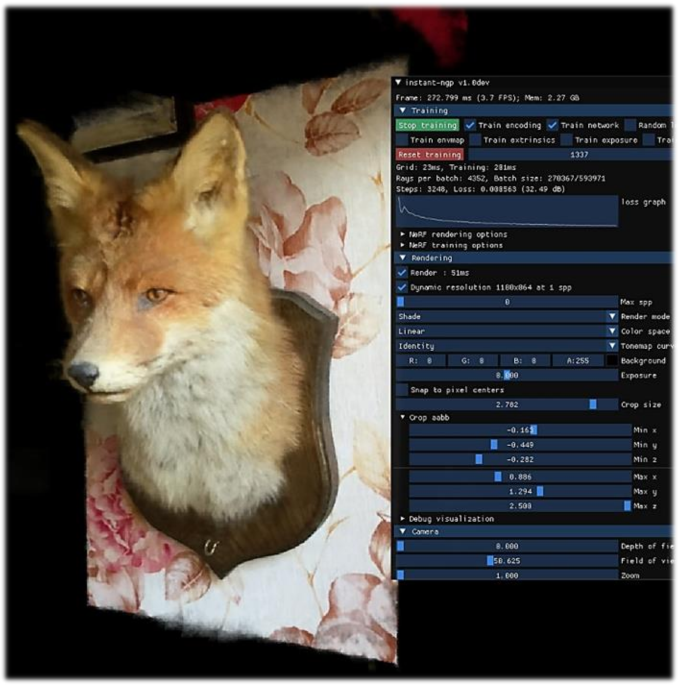
- **Image Acquisition:** Like photogrammetry, acquire a set of images of the object or scene from different viewpoints. In fact, we can work with a smaller number of images than we had in case of Photogrammetry.

- **Camera Calibration:** Calibrate the camera or imaging device to obtain accurate camera poses and intrinsics. This involves determining the camera focal length, distortion coefficients, and camera position and orientation. Usually tools like COLMAP, VisualSFM, etc., helps with these things' calibrations.

- **NeRF Training:** We then train a neural network using the acquired images to learn the 3D-structure and appearance of the object or scene. The neural network takes the 2D-images as input and outputs a 3D-density field that represents the object or scene.[3]

- **NeRF Inference:** Using the trained NeRF model, we then render novel views of the object or scene from any desired viewpoint. This involves querying the 3D-density field at different locations to obtain the color and opacity of the object or scene at those locations.

- **3D-Object Mesh Creation:** Extractin the 3D-object mesh from the NeRF model is still a work in progress, Current open-source solutions are either not that great or they are currently being used as proprietary solutions. But the best-known algorithm that is used for 3D-mesh creation is Marching Cubes algorithm. More details on the algorithm later in NeRF implementation section.



```
▼ instant-ngp v1.0dev
Frame: 272.799 ms (3.7 FPS); Mem: 2.27 GB
▼ Training
Stop training  ✓ Train encoding  ✓ Train network    Random
  Train envmap    Train extrinsics    Train exposure    Tra
Reset training                 1337
Grid: 23ms, Training: 281ms
Rays per batch: 4352, Batch size: 270367/593971
Steps: 3248, Loss: 0.008563 (32.49 dB)
                                         loss graph
 ▸ NeRF rendering options
 ▸ NeRF training options
▼ Rendering
✓ Render : 51ms
✓ Dynamic resolution 1180x864 at 1 spp
                 0                       Max spp
Shade                                  ▼ Render mode
Linear                                 ▼ Color space
Identity                               ▼ Tonemap curv
  R: 0     G: 0     B: 0     A:255       Background
               0.000                     Exposure
  Snap to pixel centers
               2.782                     Crop size
 ▼ Crop aabb
                 -0.163                  Min x
                 -0.449                  Min y
                 -0.282                  Min z
                  0.886                  Max x
                  1.294                  Max y
                  2.500                  Max z
 ▸ Debug visualization
▼ Camera
               0.000                     Depth of fie
              50.625                     Field of vie
               1.000                     Zoom
```

Fig8. NVIDIA's Instant-NGP NeRF in progress of rendering a 3D-fox(**source**)

One of the most significant advantages of NeRFs is their ability to create 3D-environments with high fidelity and accuracy. NeRFs can capture the fine details of an object's geometry and appearance, making them well-suited for applications such as virtual reality, augmented reality, and video games. Additionally, NeRFs are relatively fast, making them practical for real-time applications.

We will look into examples of current state of NeRF's 3D-model outputs and what's in the future for us later in the report. From the early days of laser scanning to the modern era of photogrammetry and NeRFs, while each of these methods has its advantages and limitations, they all represent important milestones in the development of 3D-object reconstruction. NeRFs, in particular, have emerged as a revolutionary method, enabling the creation of high-fidelity 3D-objects from 2D-images within minutes. As the field continues to evolve, we can expect to see even more exciting developments in the years to come.

3. Implementation

We are going to cover in total five different ways of creating 3D-objects- two of Photogrammetry and three that use NeRF:

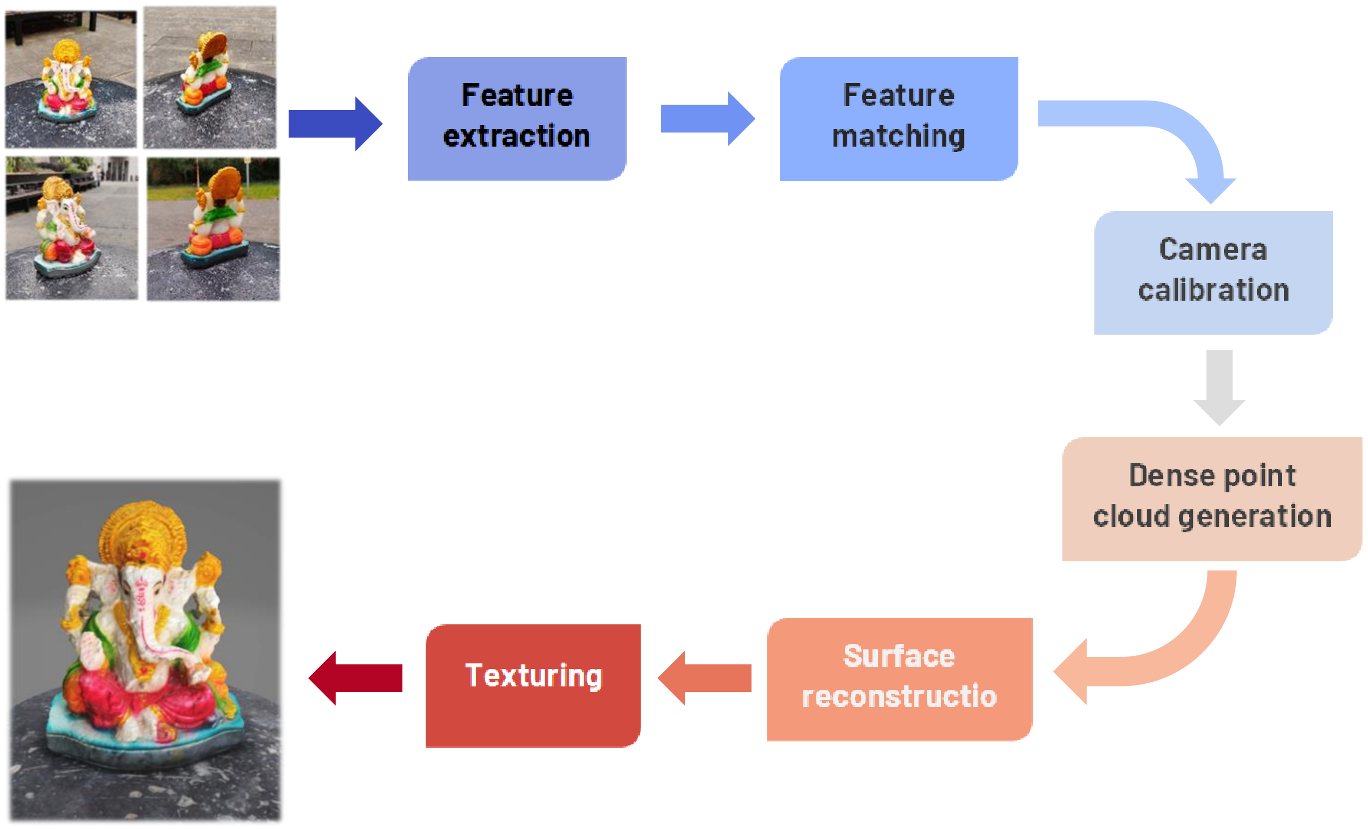
- Meshroom
- 3D-Zephyr
- NVIDIA Labs Instant-NGP
- Nerfstudio - Nerfacto
- Luma-AI

NOTE- There's guide available for each of the methods in the codebase provided along with the report.

a. Meshroom

We have seen its working before in our class assignments, so I won't go into too much details. Meshroom is an open-source photogrammetry tool, written in C++ and Python, that uses a variety of computer vision and image processing libraries to produce 3D-mesh objects using 2D pictures.

Here's Meshroom's pipeline in a nutshell:



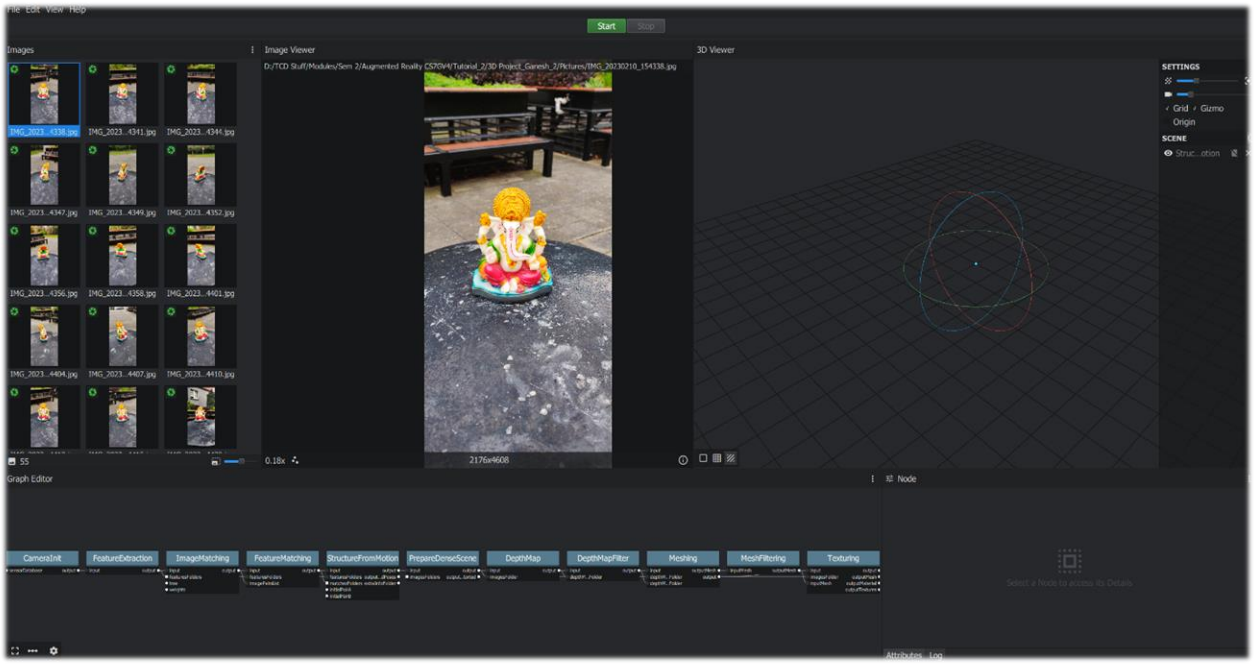


Fig9. Meshroom UI

Meshroom heavily relies on various libraries in Python and C++, we can have a small look at them below:

- **OpenCV:** An open-source computer vision library that provides a wide range of image processing and computer vision algorithms.
- **Boost:** A set of C++ libraries that provide various functionalities, including data structures, algorithms, and utilities.
- **Qt:** A cross-platform application development framework that provides a graphical user interface (GUI) for Meshroom.
- **Alembic:** A data interchange format that is used for the exchange of 3D-assets between different software applications.
- **Eigen:** A C++ library that provides linear algebra functionality, which is used for various operations in Meshroom.
- **CUDA:** A parallel computing platform and programming model developed by NVIDIA that enables the use of GPU acceleration for some of Meshroom's computationally intensive tasks.

Since we have worked on Meshroom before in our assignment and know about the whole interface and the process, so we won't go into more details here.

b. 3D-Zephyr

3D-Zephyr has been in existence since 2014 and is created by 3DFLOW. "The technology behind 3DF Zephyr has been completely in-house built by 3DFLOW, rather than relying on computer vision third party libraries…"[4]. The in-built pipeline is similar to what we saw in case of Meshroom, though 3D-Zephyr has its own few techniques that they use to create high detailed 3D-meshes. Let's discuss:

- **3DF Samantha:**

3DF Samantha is a multi-view stereo algorithm used by 3D-Zephyr to generate a dense 3D-point cloud from a set of 2D-images. It uses a combination of local and global optimization methods to improve accuracy and robustness. The algorithm works by first detecting features, such as keypoints and edges, in each image. It then matches the features between pairs of images to determine their 3D-position. From this sparse set of 3D-points, the algorithm generates a dense point cloud by interpolating the missing points.

3DF Samantha uses a hybrid approach that combines patch-based and volumetric methods. The patch-based method reconstructs small regions of the scene using a set of similar image patches. The volumetric method then fuses the patches into a continuous 3D-volume.[5]

- **JLinkage**

The JLinkage algorithm is a method for detecting multiple instances of a mathematical model from data that may contain noise and outliers. It is commonly used in computer vision and is an improvement over the RANSAC algorithm (used in Meshroom). The JLinkage algorithm is designed to be able to handle multiple models, which is a significant challenge in this field.

However, the original algorithm has a quadratic complexity on the number of data points, making it unsuitable for real-time applications. 3D-Zephyr makes use of an incremental, real-time implementation of the JLinkage algorithm that reduces its complexity and allows it to process data as it arrives. The proposed algorithm is the first to be specifically tailored for robust multiple models' estimation that can run in a very time-efficient way which helps in making 3D-Zephyr much faster in comparison to Meshroom.[6]

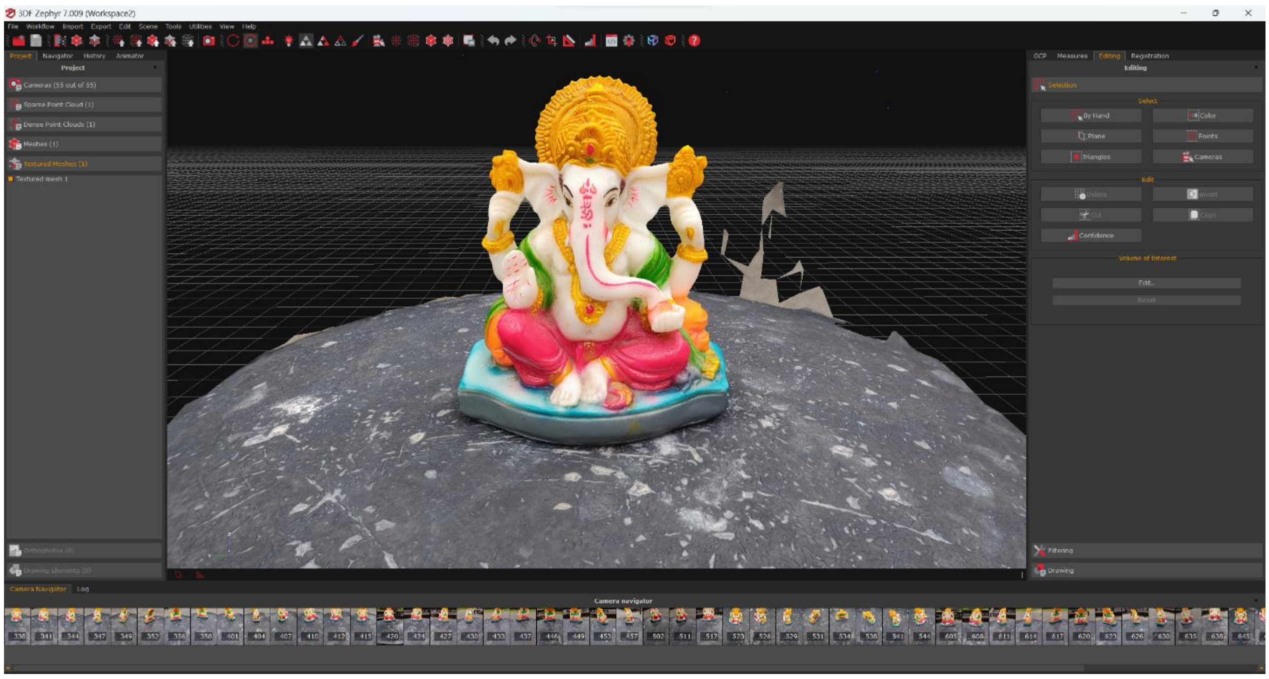
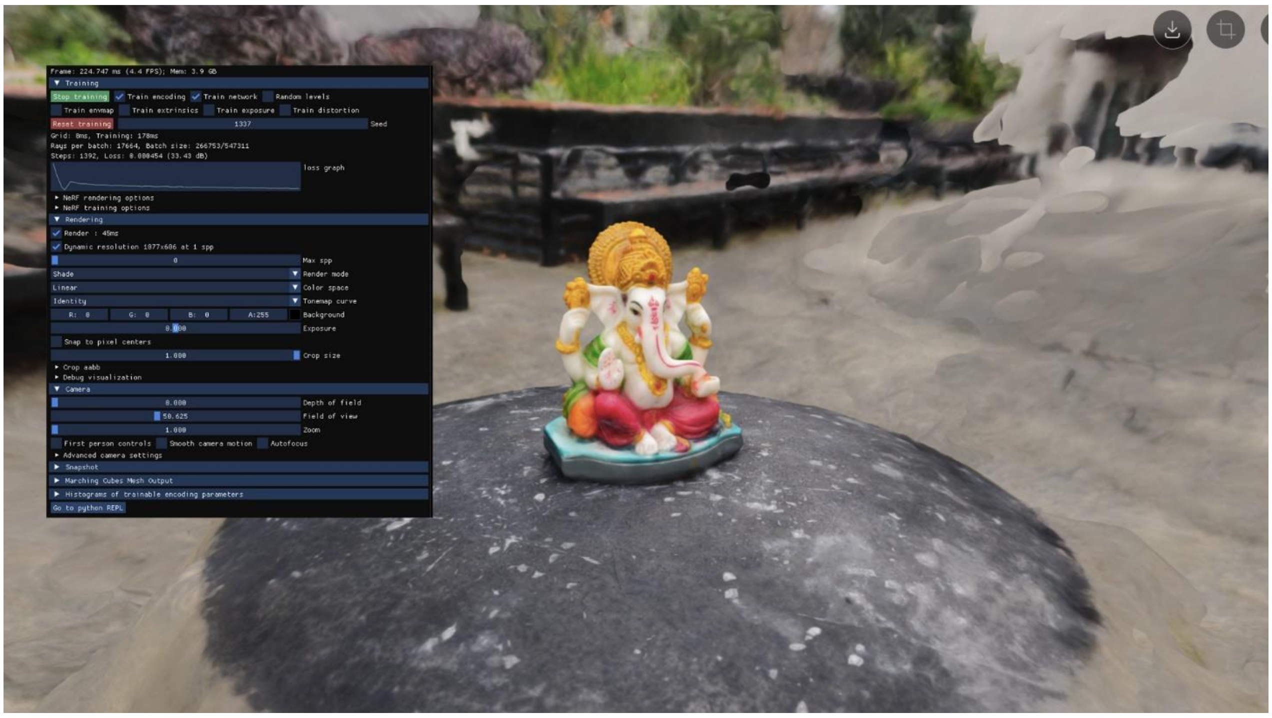


Fig10. 3D-Zephyr UI

Not much detail of how internally the tool work is available as it is a proprietary software. For this comparative notebook, 3D-Zephyr's full version was used on trial basis.

c. NVIDIA Labs Instant-NGP[7]

NeRFs have gained a lot of attention in the 3D-world since their inception in 2020. There have been multiple renditions and variations of NeRFs by now, including Nvidia's Instant-NGP - Neural Graphics Primitives, which was showcased last year. This new approach uses AI to quickly approximate how light behaves in the real world and reconstructs a 3D-scene from a handful of 2D-images taken at different angles almost instantly.



```
Frame: 224.747 ms (4.4 FPS); Mem: 3.9 GB
▼ Training
Stop training  ✓ Train encoding  ✓ Train network  Random levels
Train envmap  Train extrinsics  Train exposure  Train distortion
Reset training  1337  Seed
Grid: 8ms, Training: 178ms
Rays per batch: 17664, Batch size: 266753/547311
Steps: 1392, Loss: 0.000454 (33.43 dB)
loss graph
► NeRF rendering options
► NeRF training options
▼ Rendering
✓ Render : 45ms
✓ Dynamic resolution 1077x606 at 1 spp
0  Max spp
Shade  Render mode
Linear  Color space
Identity  Tonemap curve
R: 0  G: 0  B: 0  A:255  Background
0.000  Exposure
Snap to pixel centers
1.000  Crop size
► Crop aabb
► Debug visualization
▼ Camera
0.000  Depth of field
50.625  Field of view
1.000  Zoom
First person controls  Smooth camera motion  Autofocus
► Advanced camera settings
► Snapshot
► Marching Cubes Mesh Output
► Histograms of trainable encoding parameters
Go to python REPL
```

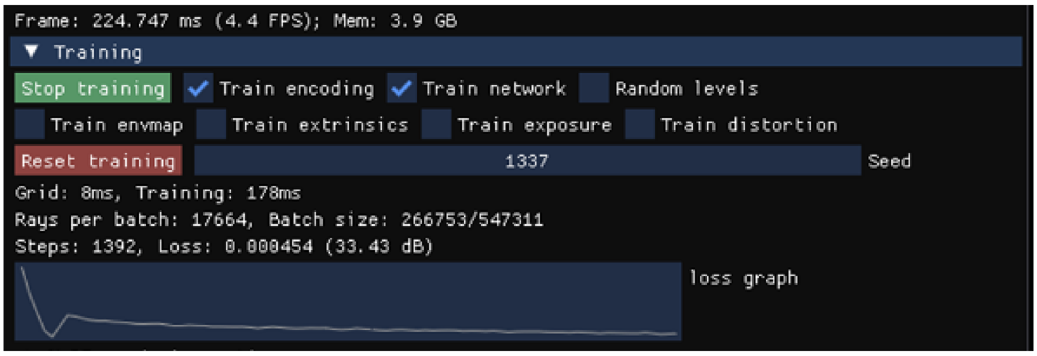
Fig11. A screenshot of NVIDIA Instant-NGP UI

The combination of fast neural network training and quick rendering makes Instant NeRF the fastest NeRF technique to date, achieving more than 1,000x speedups in some cases. With just a few dozen still photos and data on the camera angles they were taken from, the model requires only seconds to train and can render the resulting 3D-scene within tens of milliseconds. In Fig11 we can see it training on our test images of Ganesha. This above image is a screenshot after 5 minutes of training which is quite impressive.

Instant NeRF has many applications, including creating avatars or scenes for virtual worlds, capturing video conference participants and their environments in 3D, and reconstructing scenes for 3D-digital maps. But its only recently that more work has been put into turning these representations to actual 3D-objects with mesh representations. Let's briefly go through how Instant-NGP works and how do they convert this output to a 3D-mesh:

- **Data Collection:** The first step is to collect a set of 2D-images of the object or scene from different angles. The more images you have, the more accurate the 3D-model will be. Now one thing that differs in this step from Photogrammetry scenario is that when using NeRF based techniques, you are not restricted to non-shiny and non-reflective surfaces only. Such limitations aren't part of NeRF based techniques as they can handle objects with complex geometry, shiny and reflective surfaces, and even transparent materials, thanks to its continuous function representation of the scene. This makes NeRF a more versatile and powerful technique for 3D-modelling and rendering than traditional photogrammetry methods. We will look more into this through an example in the results section.

- **Ray Sampling:** Instant-NGP samples rays through the object or scene to reconstruct a 3D-volume representation. It does this by randomly selecting camera positions and ray directions from the set of 2D-images.
- **Feature Extraction:** The next step is to extract features from each ray's colour and position information, which are then used to train a neural network to approximate how light behaves in the real world.
- **Neural Network Training:** The neural network is trained to predict the colour and density of the 3D-volume representation based on the extracted features. This is done by using a loss function that compares the predicted colour and density to the ground truth values.



```
Frame: 224.747 ms (4.4 FPS); Mem: 3.9 GB
▼ Training
Stop training   ✓ Train encoding   ✓ Train network   Random levels
Train envmap   Train extrinsics   Train exposure   Train distortion
Reset training   1337   Seed
Grid: 8ms, Training: 178ms
Rays per batch: 17664, Batch size: 266753/547311
Steps: 1392, Loss: 0.000454 (33.43 dB)
```

Fig12. A screenshot of Instant-NGP's model training

- **Rendering:** Once the neural network is trained, it can be used to render the 3D-model from any viewpoint. To do this, Instant-NGP samples rays through the 3D-volume representation from the desired viewpoint and uses the neural network to predict the colours and density of each ray.

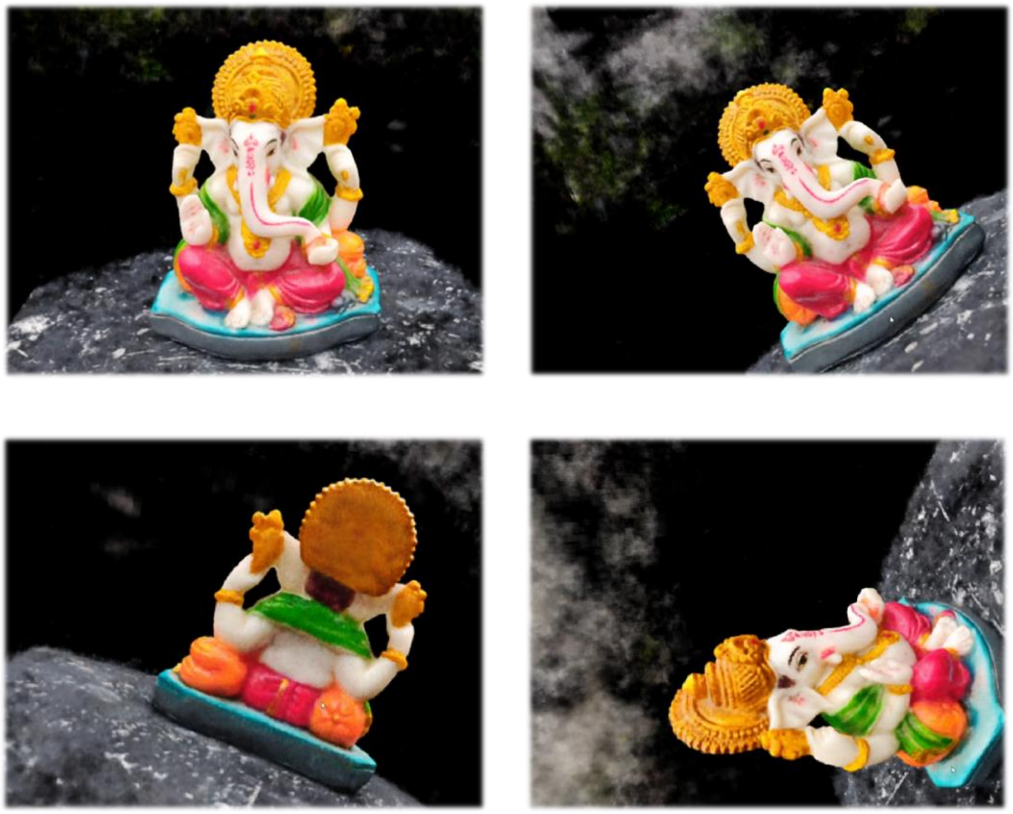


Fig13. Real time 3D-renders from a noise cropped environment using Instant-NGP

- **Optimization:** To improve the accuracy and speed of the rendering process, Instant-NGP uses an optimization technique called hierarchical importance sampling. This allows the model to focus on the most important rays while ignoring the less important ones. This is the reason some parts of above image might be looking with lot of aerial artifacts.
- **Output:** The final output is a high-quality 3D-model that can be rendered from any viewpoint in real-time. This model can be used for a wide range of applications, including virtual reality, video conferencing, and 3D-digital mapping.
- **Mesh generation:** Instant-NGP NeRF does not generate meshes directly. Instead, it generates a 3D-volume representation of the scene or object, which can then be converted into a mesh using a technique called Marching Cubes[8]. Marching cubes is a well-established technique in computer graphics that converts a 3D-volume representation into a surface mesh. It works by dividing the 3D-volume into small cubes and examining the density values of each corner of the cube. Based on the density values, it then determines which edges of the cube intersect the surface of the object and generates triangles along those edges to create a mesh.

NVIDIA's Instant-NGP is available as an open-source solution on **GitHub**.

d. Nerfstudio Nerfacto[9]

Nerfstudio is again an open-source project ran by a group of Berkeley students, who have come together to form easy to follow and accessible tool to create 3D-renders of images using different NeRF methods. This tool has implemented 7 different types of NeRF models including Instant-NGP from Nividia. We are going to talk about one of these methods- Nerfacto.

Nerfstudio has developed Nerfacto model, which combines several techniques from existing published methods to improve the performance of the Neural Radiance Fields (NeRF) algorithm for real data captures of static scenes.

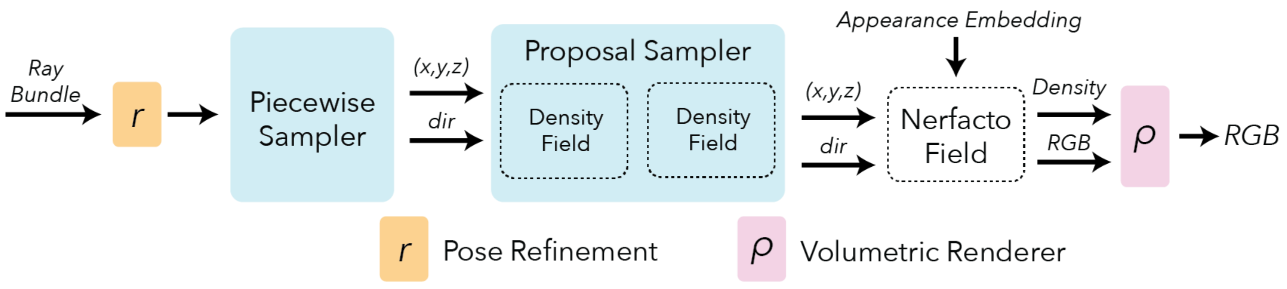


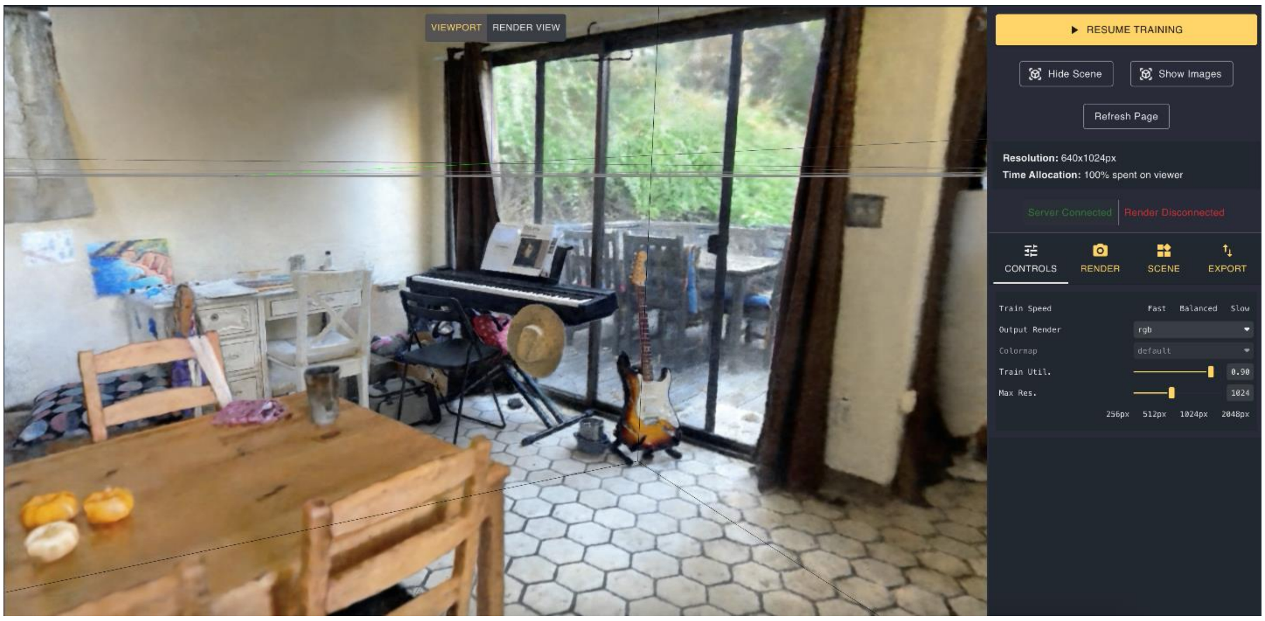
Fig14. Overview of pipeline for Nerfacto**(source)**

Here are the improvements that Nerfstudio has made to the existing NeRF:

- **Camera pose refinement:** Errors in predicted camera poses can result in cloudy artifacts in the scene and a reduction of sharpness and details. Nerfacto uses the NeRF framework to backpropagate loss gradients to the input pose calculations and optimize and refine the poses.
- **Piecewise sampler:** Nerfacto uses a piecewise sampler to produce the initial set of samples of the scene. This sampler allocates half of the samples uniformly up to a distance of 1 from the camera and distributes the remaining samples such that the step size increases with each sample, sampling distant objects while still having a dense set of samples for nearby objects.
- **Proposal sampler:** The proposal sampler consolidates the sample locations to the regions of the scene that contribute most to the final render, which greatly improves reconstruction quality. The

proposal network sampler requires a density function for the scene, which can be implemented using a small fused-mlp (multi-layer-perceptron) with a hash encoding, providing enough accuracy & speed.

- **Density field:** The density field only needs to represent a coarse density representation of the scene to guide sampling. Combining a hash encoding with a small fused MLP provides a fast way to query the scene, and simplifications such as decreasing the encoding dictionary size and number of feature levels have little impact on the reconstruction quality.



Resolution: 640x1024px
Time Allocation: 100% spent on viewer

```
Train Speed      Fast  Balanced  Slow
Output Render    rgb
Colormap         default
Train Util.      0.90
Max Res.         1024
     256px  512px  1024px  2048px
```

Fig15. Nerfacto viewer-UI**(source)**

Rendering outputs are similar to Instant-NGP, so we will just move on to last implementation and see the final results of Nerfacto in result section.

e. Luma-AI

Found in 2021, Luma-AI has gained widespread attention as an app and service developed by Luma Labs that uses Neural Radiance Fields (NeRF) to capture 3D-images. They provide an easy-to-use platform in form of a website and mobile application that allows users to upload images or videos and then convert them into 3D-models. Unlike other tools that require users to handle technical tasks such as cloning repositories or installing libraries, Luma-AI takes care of the heavy lifting, allowing AR and VR professionals to easily create 3D-objects from images and incorporate them into their work.

Luma-AI is a proprietary software, and there isn't much information available on how it works. It's worth noting that the 3D-mesh object that Luma-AI creates is much more impressive than the results obtained from other tools such as Instant-NGP and Nerfacto. When asked about their process on Discord (provided in the appendix), the original developers provided limited information, with the closest clue being that they use some variation of Marching Cubes.

Here are the results of using Luma-AI to create NeRFs:

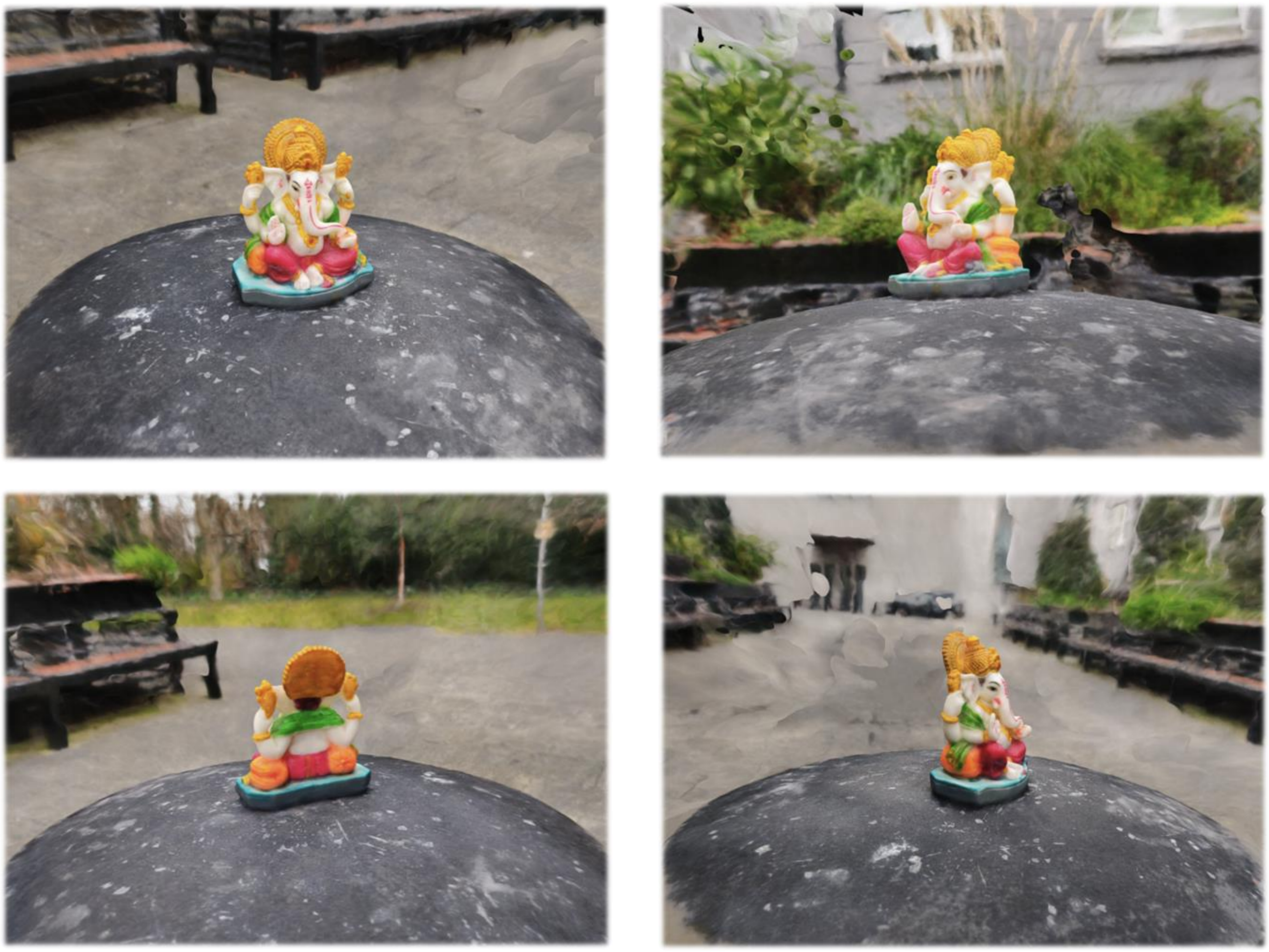
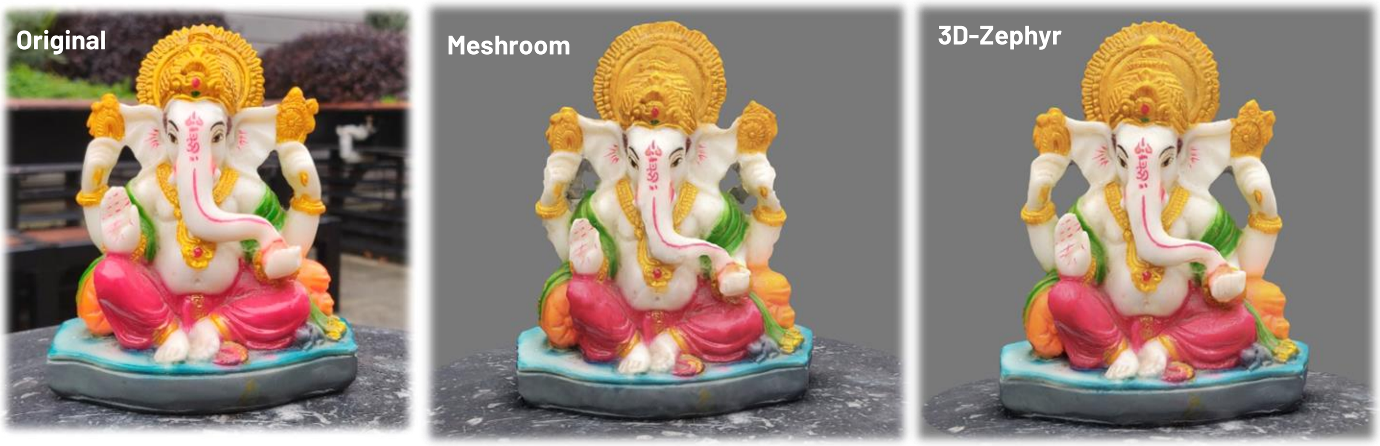


Fig16. 3D rendered outputs of NeRF via Luma-AI from different angles

4. Results

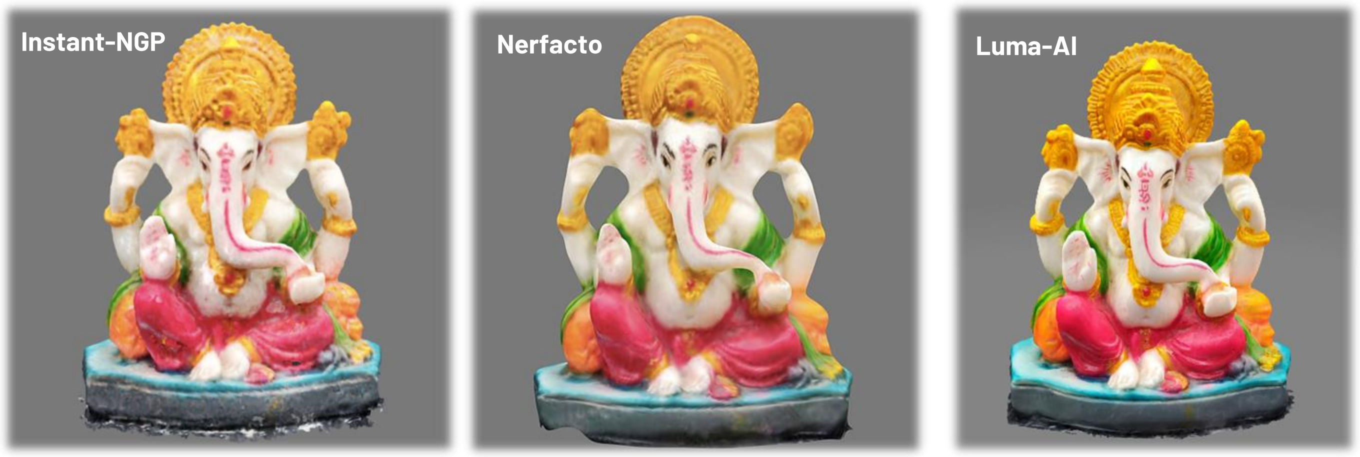
It can be seen below that the results from Photogrammetry are really great in quality, high resolution and lot better than current NeRF based solutions except Luma-AI. NeRFs are fairly new and they aren't customized to be used for 3D-Object Mesh Reconstruction but there is some hope as players like Luma-AI are joining the field with great results to showcase.

Here are the results from two photogrammetry software vs one of the original sample images:



A beautiful thing about 3D-reconstructions is that once created, we can set up the 3D-object in any angle we want and thus I was not able to find any sample image that had same view. We can see how clear, crisp and high-resolution results were obtained using Meshroom and 3D-Zephyr. 3D-Zephyr gives a really well-suited free alternative to Meshroom. Moreover, it is able to construct the sample image much more precisely.

And here are the results from NeRF based solutions. Clearly Luma-AI has edge over Instant-NGP and Nerfacto. In fact, LUMA-AI's 3D-mesh output looks better and closer to the original image than what we got from Meshroom and 3D-Zephyr.



Here are some mesh objects related information for each model and right away one striking thing to notice is the material count for Luma-AI. The logic behind this is that for a 3D-object, the outer object shows colour using a texture image. Now for all the other methods, the texture material is just 1 single image. But for Luma-AI output we have got 84 different materials which together form the final image. This means that internally at the backend they are making the 3D-meshes in parts or performing this operation recursively and thus generating multiple materials, which gives a more precise output and this would work out quite well for small object like ours but might not be that effective in case of large scenes as also mentioned by one of the users in their blog.

Method	**Triangles**	**Vertices**	**Materials**
Meshroom	424.5k	212.9k	1
3D-Zephyr	152.9k	76.8k	1
Instant-NGP	676.1k	339.7k	1
Nerfacto	1.9M	926k	1
Luma-AI	811.9k	571.4k	84

All the four models except Luma's are available as objects in Sketchfab while Luma-AI's object can be accessed in the website itself. These models are also uploaded as 3D-meshes in the google drive and can be downloaded and used for future purpose.

Now we were working with well lit but less reflective surfaced images and it gave really good results. But we have seen before from our Meshroom assignment that if trying to recreate an object with lot of reflections, the results does not come out to be that good and there's a reason for that. While doing the research for this project I stumbled upon this example:

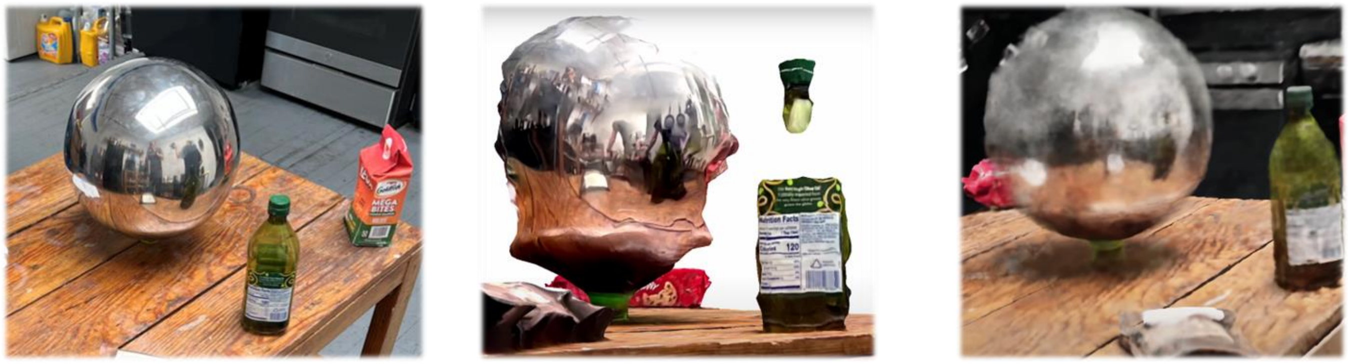


Fig17. Original image vs Polycam(photogrammetry) vs Instant-NGP

These images are taken from experiments performed by Wren Weichman and from videos published by him.[10] [11] He used same images to produce 3D-outputs using Polycam that used Photogrammetry and Nvidia Instant Nerf. We can see how in second model image output, photogrammetry has not been able to do well at all, whereas Instant-NGP is able to work much better if not worse. It is able to recognize the ball as a ball with reflections and also the transparent bottle is intact. Now this is a great example of why NeRF might be able to handle things better in lot more cases and should be looked more into. So why is it like that?

Why do NeRFs work and are able to handle reflections better than Photogrammetry?

Photogrammetry is a technique that uses multiple images of an object or scene from different angles to reconstruct a 3D-model. This is done by analysing the pixel information in each image and triangulating the corresponding points in 3D-space. However, photogrammetry relies on consistent pixel information across all angles. This means that if an object has reflective or transparent surfaces, the pixel information in each image may be inconsistent, leading to errors in the reconstruction. This is exactly what is happening in case of Chrome ball in above image. In fact, transparent objects fare in same way and due to similar reason.

NeRF, on the other hand, is a deep learning-based method for synthesizing novel views of a scene from a set of input images. It works by predicting the radiance (colour & opacity) at any point in 3D-space using a neural network. The main idea behind NeRF is to represent the scene as a continuous 3D-function, where the function values are the radiance values at each point. This continuous function representation allows NeRF to handle objects with complex geometry, shiny & reflective surfaces, and even transparent materials. The neural network used in NeRF is trained on a set of input views of the scene, and it learns to predict the radiance values at any point in 3D-space depending on the viewpoint. This allows NeRF to render novel views of the scene from any viewpoint, with accurate colour and lighting information.

Here's another example from Wren, but this time it was Polycam vs Luma-AI. And here are the results:

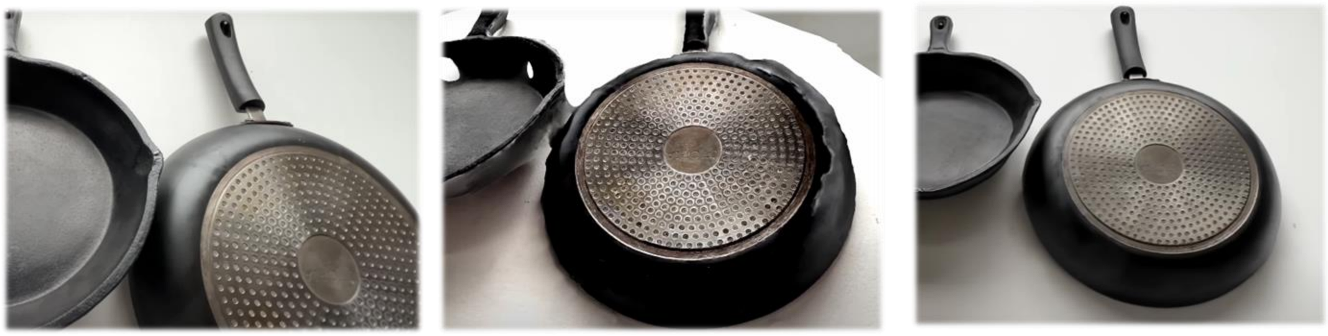


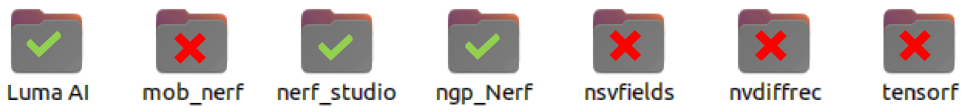
Fig18. Original image vs Polycam(photogrammetry) vs Luma-AI

And as we saw in our own results Luma-AI results are so realistic and mind-blowing that it is hard to tell the difference between original image vs 3D-reconstruction.

5. Summary, Conclusion and Future Work

It's no surprise that Photogrammetry gives great results and helped developers to use any object's 2D-images and create reusable 3D-object out of it but it has its limitations. And even though NeRF based solution aren't mainstream right now and readily available, it's indeed exciting to see what we might see in near future. In fact, last year only Nvidia brought an update to their Instant NeRF & created Nvidia MoMa, a method that can use neural network-based methods to recreate 3D-objects from images.**[12]**

It is also available as a Github repository but I was not able to make it work. In fact, I attempted seven different methodologies for NeRFs, but unfortunately, not all of them were successful.



Some of these approaches demanded extensive customization and efforts, while others had dependency issues with several other libraries. Although some of these techniques are fairly new and have been praised for their ability to recreate 3D-mesh objects, I struggled to make them work. However, I did get a new insight which might be of some value from these additional experiments- providing masked segmentation of the object we want to create a mesh for to the mode, significantly improves the final 3D-mesh objects.

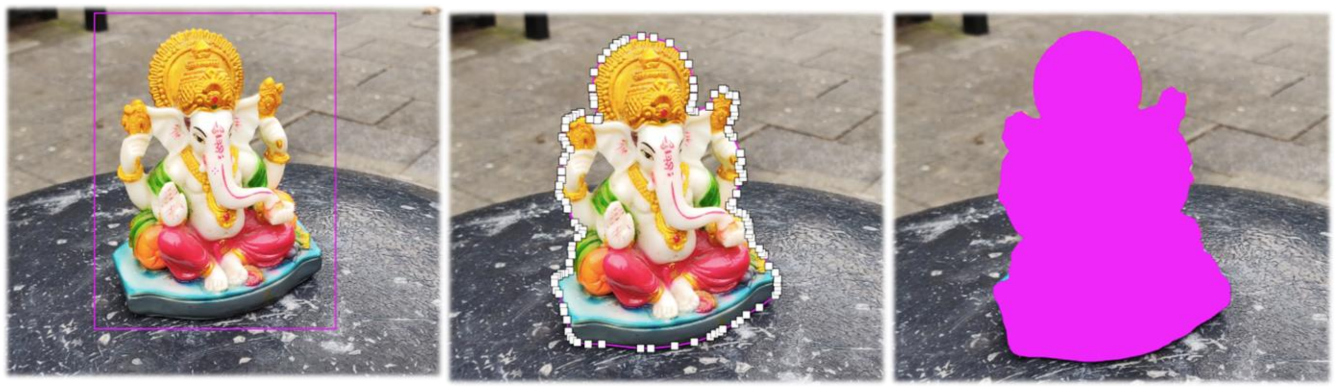


Fig19. Example of Mask of the Ganesha object in our use case

Almost all the new methods required mask segmentation along with the training images which I can only assume helps the NeRFs to create much better looking meshes. All these are really new and hopefully in near future we will have lot better results. Here are links to the above-mentioned methods which didn't work if anyone in future would like to work with them:

- NVIDIA MoMa (nvdiffrec)
- MobileNERF
- TensoRF
- NR Fields

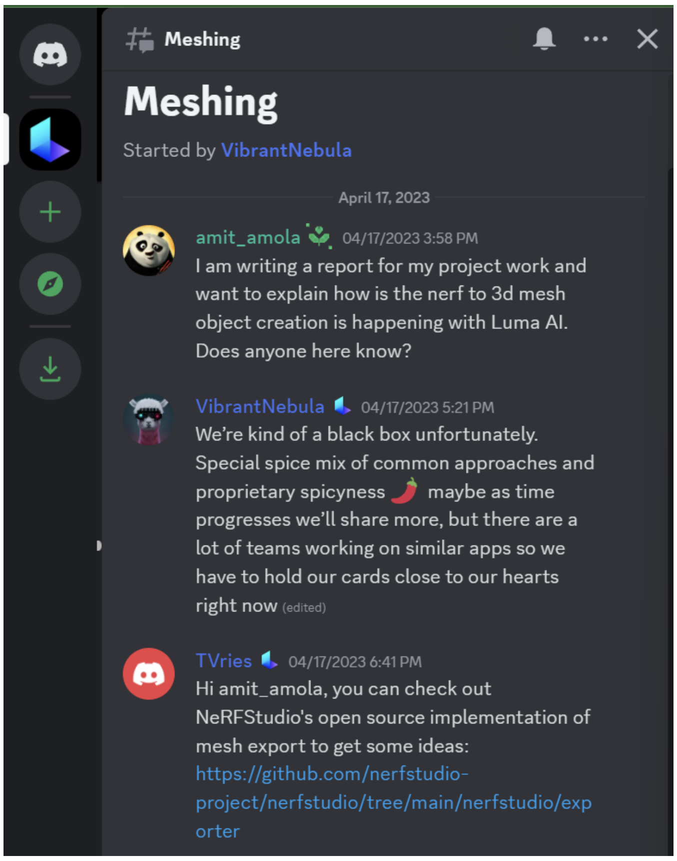
Finally, here's the link to the demo which goes through the report in general along with some commentary on what the project is all about: https://www.youtube.com/watch?v=P-U7F5AvmAU

Reference

1. Geoffrey Jacobs, "The Early Days of 3D-Scanning - Part 6," XYHT - A Magazine for Geospatial Professionals, Nov. 2020-**(link)**
2. J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion Revisited," in Proceedings of the IEEE Conference on (CVPR), 2016-**(link)**
3. Sitzmann, Vincent, Michael Zollhöfer, and Gordon Wetzstein. "Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations." arXiv-**(link)**
4. 3D-Zephyr wiki-**(link)**
5. Toldo, Roberto, Riccardo Gherardi, Michela Farenzena, and Andrea Fusiello. "Hierarchical Structure-and-Motion Recovery from Uncalibrated Images-**(link)**
6. Toldo, Roberto, and Andrea Fusiello. "Real-Time Incremental J-Linkage for Robust Multiple Structures Estimation," September 2010-**(link)**
7. Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," 2022-**(link)**
8. Lorensen, William E., and Harvey E. Cline. "Marching Cubes: A High-Resolution 3D-Surface Construction Algorithm." ACM SIGGRAPH Computer Graphics-**(link)**
9. Nerfstudio Nerfacto-**(link)**
10. Wren Weichman's tweet about NeRFs-**(link)**
11. Wren Weichman's youtube video about NeRFs-**(link)**
12. NVIDIA MoMa-**(link)**

Appendix

❖ Lumalabs developers' response



Meshing

Meshing

Started by VibrantNebula

April 17, 2023

amit_amola 04/17/2023 3:58 PM
I am writing a report for my project work and want to explain how is the nerf to 3d mesh object creation is happening with Luma AI. Does anyone here know?

VibrantNebula 04/17/2023 5:21 PM
We're kind of a black box unfortunately. Special spice mix of common approaches and proprietary spicyness 🌶️ maybe as time progresses we'll share more, but there are a lot of teams working on similar apps so we have to hold our cards close to our hearts right now (edited)

TVries 04/17/2023 6:41 PM
Hi amit_amola, you can check out NeRFStudio's open source implementation of mesh export to get some ideas: https://github.com/nerfstudio-project/nerfstudio/tree/main/nerfstudio/exporter