ELEC4848 Senior Design Project 2024-2025

Abel Haris Harsono (3035834345)

3D Scene Reconstruction for Simulation Using 3DGS/NeRF

Supervisor: Prof. Y. Wang
Second Examiner: Prof. Xiaojuan Qi

Abstract

Recently, the ability to take a set of images or videos of a particular scenery to produce its 3d reconstruction has gained popularity. Works such as Neural Radiance Field (NeRF) and 3D Gaussian Splatting (3DGS) contributed significantly to this trend and made ways for further research within the field. One aspect in particular deals with large scene reconstruction. While numerous works have been done, the approaches are typically complicated and either require the whole scene to be captured first, or they struggle as the scene gets bigger. This project aims to address this gap by proposing a different approach that chains reconstructions of smaller scenes to form the bigger scene. The project presents an implementation of the approach using COLMAP and shows some reconstructed scenes to demonstrate its capabilities. One of the reconstructions include a 3D point cloud of The University of Hong Kong's Innovation Wing 1. The report also shows the usefulness of exploiting neighboring information and how chaining can help achieve this. The method is not perfect however and may struggle as the scene gets bigger due to errors in the previous reconstructions among other practical limitations.

Acknowledgment

I would like to thank Prof. Y. Wang and Prof. Xiaojuan Qi for their guidance and to allow for the proposal of this final year project. I would also like to thank Dr Yang Lei for his comments and The University of Hong Kong's Tam Wing Fan Innovation Wing for allowing me to scan rooms within Innovation Wing 1. Finally, I'd like to thank the PhD students of the Computer Vision and Machine Intelligence lab for their advice while working on this project.

Contents

List of Figures

Abbreviations

3DGS 3D Gaussian Splatting

DBMS Database Management System

HKU The University of Hong Kong

Innowing The Univeristy of Hong Kong's Innovation Wing I

MiB MebiByte

NeRF Neural Radiance Field

1 Introduction

1.1 Background Information

The idea of taking videos or images of a real life scene to reconstruct the captured 3D scenery has received attention as of late. Typically, images or videos, the 2 dimensional representations of a scene, are produced from 3 dimensional models. These 3D models can be 3D meshes or even the physical world we live in. The idea goes in the opposite direction of that process by recovering the 3D structure of the scene from its 2D representations. This is often termed as 'inverse graphics'. The first pieces of work that popularized it was NeRF [1] and followed later by 3DGS [2], a notable improvement of NeRF.

The availability of tools that ease the processes of inverse graphics brings about new research possibilities e.g. Neural texturing [3], scene segmentation [4], and 3D object generation [5] to name a few. Of these great topics, one in particular pertains to not just recreating a scene but also making the reconstructed scene interactive by either recovering or adding a predefined mechanisms to the reconstruction. Basically, it aims to recreate a world along with some mechanics, or in other words, to simulate.

Video2Game [6] is a great example within the topic. It can convert a scene captured with NeRF into a playable video game level. The paper also presents different game modes, collision with objects within the world, and a simulated scene with robots as players. Additionally, predictions of in-game object properties using language models were also discussed. Overall, a great work in combining different techniques together to make the reconstructed scenery interactive.

1.2 Problem Definition

There are many works within the literature that can be utilized for the purposes of simulations. One of particular interest deals with the processing of big scenes. Before moving forward, for context, the process of inverse of graphics typically involve two stages: the initial 3d reconstruction stage and the rendering stages where the initial reconstruction are used as input. Dealing with large scenes can be done at both stages.

Works by Zhu et al. [7] and Qu et al. [8] for examples, deals with the problem at the initial reconstruction stage. Zhu et al. Developed a way to cluster cameras together so that the reconstructions can be done on each cluster and merged afterwards while Qu et al. Proposed an algorithm to better choose key images.

Within the later stages, there are exemplars in the likes of Hierarchical Gaussian Splatting [9] and BlockNeRF [10]. BlockNeRF works by splitting a scene into smaller components and store each component within a NeRF. A NeRF in essence is a neural network. To use BlockNeRF then, the neural networks along with their weights need to be saved. To view the scene or its components, inferences are made through the networks. Hierarchical Gaussian Splatting follows a similar process but swaps out NeRF with a 3D gaussian model and introduces a way to describe hierarchy between the models.

While these approaches work great and can produce remarkable results, they are also quite complicated and have their own set of imperfections. BlockNeRF, since it uses NeRFs, can be expensive to infer through as neural networks are usually slow and uses a

lot of memory to run. Hierarchical Gaussian Splatting on the other hand, even though it improves upon BlockNeRF by not making use of any neural networks, can still suffer from visual artifacts in the scene rendering due to bad input data. Works by Zhu et al. and Qu et al. falls into a category named 'Incremental SfM' and works from there may suffer from bad initial images, accumulation of errors, and efficiency problems [11]. Furthermore, all the mentioned works either require the whole scene to be captured before breaking it down or may eventually struggle as the scene gets bigger. A possible direction to think about then is on the possibility of achieving splitting of reconstruction in an efficient and simple manner without needing the entire scene to be present and to not use all of the already reconstructed components at once when new parts of the scene is captured. This final year project concerns itself with the possibilities of doing such work in the initial reconstruction stage with tools like COLMAP [12].

1.3 Importance

All things considered, the importance of the project lies in proposing a different approach in dealing with bigger scenes that would allow for better performance and convenience. Performing the splitting earlier could bring improvements in terms of speed and memory usage because it allows for the big scene to be processed one subset at a time. Having less scenes to work on lead to less images being processed which eventually could mean less memory use and shorter time for reconstruction.

Splitting at this stage would also allow for simplicity and convenience. The approach taken in this project has little added complexity and mainly works on top of present and arguably well known tools which made simpler system designs possible. Additionally, since only subsets of a scene is needed, management of different scenes during its procurement could also be somewhat improved as now, the concern can be fixed to some, possibly small, boundary in space. Not all of the components are required to be available during the splitting process as well and can be captured later.

1.4 Objectives & Deliverables

The goal of the project is to develop an extension for COLMAP using python that will allow for scene splitting and achieve the previously mentioned benefits. A visualizer system, viewable from a browser, will also be developed to present the results of the reconstruction process. As to show the capabilities of this approach, the project presents an attempt to create a 3D reconstruction of The University of Hong Kong's Innovation Wing 1. It is a complex scenery with different levels and many small details contained within.

1.5 Outline of The Progress Report

This progress report will first go through the methodology in section 2 where the system architecture and designs are discussed along with their motivations (section 2.1, section 2.2, section 2.3). Section 2.4 talks about a visualizer system for the project and section 2.5 discusses about an algorithm to help improve reconstructions. Section 3 shows the results of the reconstructions made with the proposed idea. It also presents analyses and arguments for the reconstructions along with the overall approach (section 3.2.1 - section 3.2.5). After that, section 4 lists some limitations to the current approach and section 5 describes challenges faced during the working of the project. Finally, section 6 will conclude the report.

2 Methodology

This section shows the important parts of implementing the idea proposed in this project. It goes through the abstractions and the main algorithms used by the approach along with the resulting system designs. The visualizer system is also described in this section.

2.1 Problem Abstractions

This project abstracts the notion of a scene. That is, a scene, a room, multiple rooms, multiple scenes, big or small scenes, are all the same thing; they are all a set of overlapping images. For convenience, those types of terms will be used interchangeably throughout this report. At times, the term 'model' may also be used instead. This abstraction is made due to images being the fundamental quantity representing a model and is typically an input, sometimes the only input needed, to a framework that produces 3d reconstructions.

Another abstraction used regards to how the problem is viewed. As mentioned, the project aims to perform splitting of a scene during its early reconstruction stages. At this point, there is not much to work with other than the point clouds of the reconstructions. If the scene is to be broken down into smaller parts, this would mean that a significant part of the implementation deals with how the reconstruction connects with one another. Since the problem mostly deals with connectivity of different objects, one way to deal with it is to use graphs. A graph is a convenient and well known structure made up of nodes and edges. They are typically used to model problems which involves entities having relationships with one another; the nodes represent entities and the edges for the relationships. For the problem this project is concerned with, the nodes are the models and the edges indicate model overlaps. An important thing to note here is that the project only deals with connected models i.e. The graph will always be a connected graph.

2.2 System Architecture

2.2.1 Platform Setup

This project utilizes python and COLMAP. First, COLMAP is a well known framework for doing Structure From Motion and Multi View Stereo. It takes in as input a set of images, computes their correspondence, and outputs point clouds of the 3D scenery captured by those images. COLMAP is very well known within the literature and beyond. It has been used to accompany a multitude of recent works despite its age. Even though there are other newer, more efficient works, the project will still utilize COLMAP. This decision is due to its maturity from having been published for quite some time. A considerable amount of issues and functionalities are more likely to be solved and provided with COLMAP's code base than others. Second, python will be used to develop the main application on top of COLMAP utilizing pycolmap; a python package for COLMAP. It has some limitations e.g. The inability to use CUDA to do dense reconstruction, restrictions to modify the source code, etc. But those limitations are not so relevant to this project. Nothing fundamental of COLMAP will be changed and the project only aims to use what has already been provided. Given the circumstances, the use of python with pycolmap would be more beneficial and productive.

2.2.2 System Design

In order to solve the problems defined in section 1.2 with only having information on images and their inherent geometry, one possible way is to reconstruct the models one by one. To explain this, first begin with a big house with many rooms. Typically, many images of the many different rooms within the house will be taken and fed directly into COLMAP. An alternative is to first reconstruct any room within the house. Then, pick a second room that is connected to the first, reconstruct the second room jointly with the first but keep the first reconstruction fixed in place. If the images of the rooms are separated, the second room's reconstruction can be extracted from the combined reconstruction. Doing so would lead to two separate reconstructions of different rooms. This process can be continued until all rooms within the house are reconstructed; for the next iteration, simply make the second room act like the first room. Additionally, since the reconstructions were at one point joined, the alignment information can also be extracted and saved during that moment. The project termed this approach as 'reconstructions chaining'. In terms of graphs, the approach is essentially to fix the reconstruction of a neighboring node of the target node while the target is being reconstructed. Figure 1 illustrates this chaining approach in the case of two scenes.

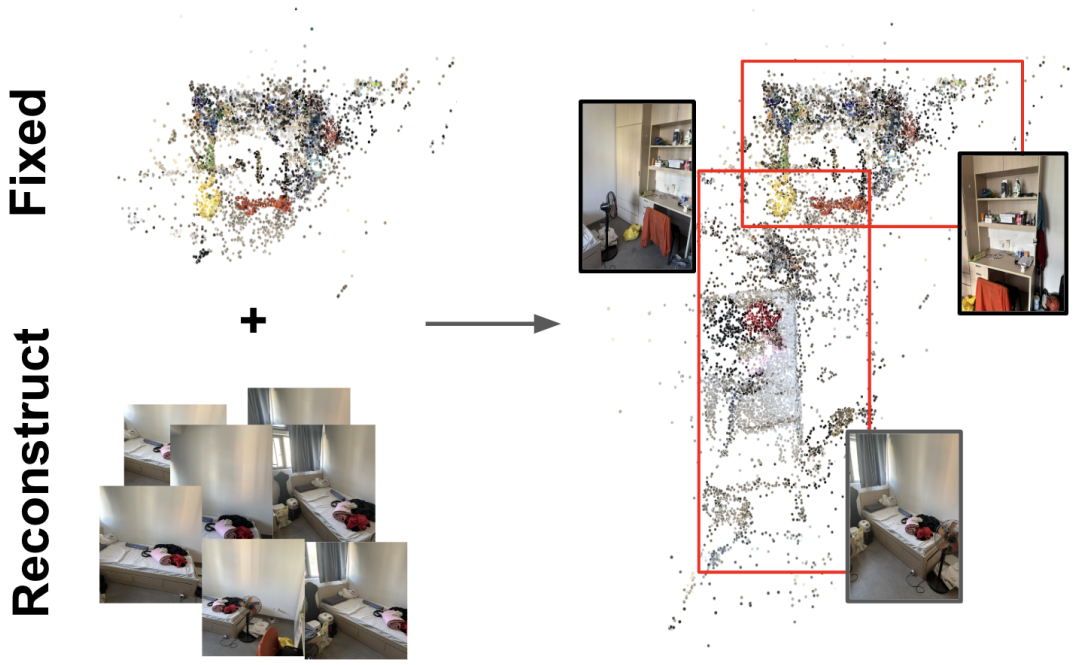


Figure 1: Reconstruction Chaining

An alternative to this approach is to reconstruct each model independently but ensure they share some images in common. An issue with this is with how multi-view geometry is done. The scale of the overall scene is unknown. It is quite arbitrary and with respect to what COLMAP was intended to do, it indeed is. But to split the scenes, the scale information becomes important. Furthermore, COLMAP tends to modify the final size of the scene and their position by normalizing it. This is done for better numerical stability. The chaining aspect introduced here is to take care of these sorts of problems by

providing a reference. Basically, now different reconstructions are forced to share these arbitrary values. This is also why the first reconstruction is fixed in the approach described.

The idea presented here motivates the choice of splitting the application into different services: Services for dealing with chaining reconstructions and services to extract alignment information. The overall process also involves a significant amount of file and database input-output. For instance, chained reconstruction service may need to save the reconstruction into a folder or to update whether a model has been reconstructed. As such, the application also has three more services: file services, database services, and data services. Data services is here to accommodate when file services and database services are required at the same time. Their overall dynamics is shown in figure 2 where an arrow from one box to another means that box uses the services of the other box. Put it simply, both the alignment service and chained reconstruction service both utilize the input-output related services. The chained reconstruction service makes use of the alignment service as described before.

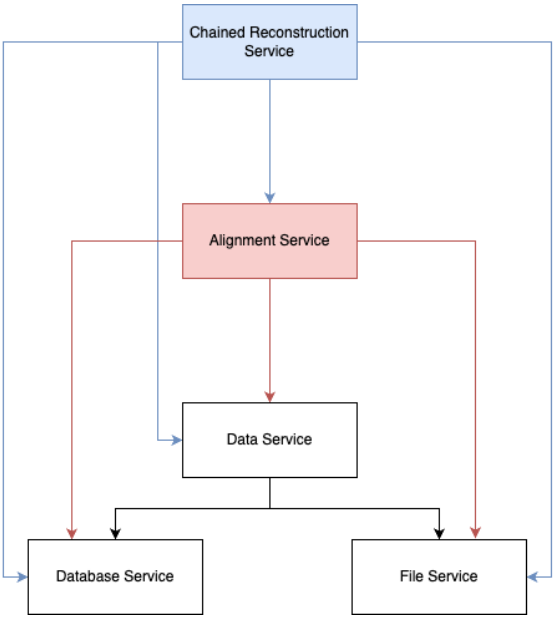


Figure 2: Components Relationship

2.3 Database Design

2.3.1 Database Setup

The system utilizes a database system to store information on registered models, alignment information, and neighboring information. Each of those was put as a table within the database. These different quantities are required by the project because they contain the graph representation of the problem (see section 2.1). A database here is used for convenience since the amount of data to be stored is not bounded and the modification of certain values may happen frequently. The database systems would also allow for more efficient operations out of the box. Furthermore, with features usually provided with databases, future improvements of the project e.g. Concurrency, adding more quantities, etc. would be simpler. The DBMS picked here is MySQL. It is straightforward to use and has a friendly user interface. It is also quite popular which mean there are plenty of resources available to support projects that utilizes it. Figure 3 shows the relations between the different tables within the database.

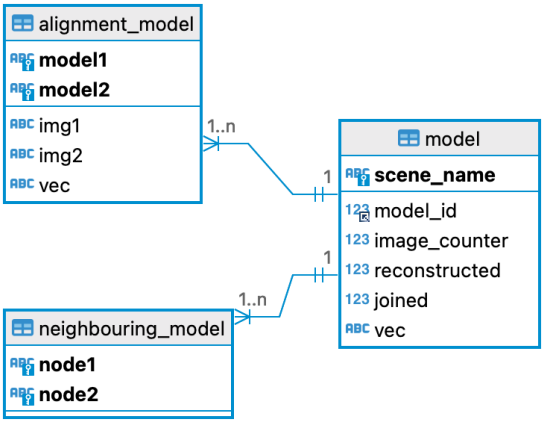


Figure 3: ER Diagram

2.3.2 Columns Descriptions

The following describes what the columns of the tables mean and what they do. First, the term 'model' (from alignment_model) and 'nodes' (from neighbouring_model) mean the same thing. In fact, they are foreign keys (a reference) from the model table's primary key scene_name which just gives the name for a particular model/node. Then, the img columns represent images associated with a particular model. So, img1 would refer to model1's image. The usage of the img columns relate closely to the vec columns to be explained shortly. After that, for the model's table, 'reconstructed' indicates whether the model has been reconstructed and image_counter is used to keep track of how many images are associated with a model. The 'joined' column indicates whether or not the particular model has been joined to a mesh. This is to support joining of the smaller models to form the big scene.

Finally, the vec column stores 3D vectors and has different roles in different tables. In the alignment_model table, vec stores information of the difference of vector between img1 and img2 which is how the alignment information is extracted and stored. Storing the difference of vector between images is a straightforward approach to record alignment since

COLMAP stores the information of the location of the images within a reconstruction. Please refer to COLMAP's documentation for more details.

The vec column in the model table stores where the model has been translated in the big reconstruction. This column is used during the process when all the smaller reconstructions are being combined. To see why vec is useful here, it is helpful to begin with three reconstructions labeled A, B, and C with A being neighbors with B and B with C. Joining A and B is straightforward as A can be taken to have it's position fixed and since their relative positions are recorded (vec from the alignment_model table), B can be translated by that amount. The issue is when C is to be joined. The relative positions between B and C are known but the current whereabouts of B is unknown and hence C cannot be placed properly. While position B can be known if the current joined reconstructions are labeled, doing this efficiently may not be straightforward. The project takes a simpler approach of saving a translation vector instead. As such, when the position of the reconstruction in the the big reconstruction is needed, simply take the position of the saved reconstruction and add vec.

Note that it is also possible to just move the original reconstruction and forgo the use of vec. But, it would be inefficient as a bigger input-output cost would be incurred. This is a comparison of saving a reconstruction that may contain thousands of 3D points (basically, thousands of vec) against storing a single vec.

2.4 Visualizer

This project includes a visualizer developed with Three.js to view the reconstructions. Three.js is a popular library for doing WebGL or in other words to do 3D graphics on the web. A web based graphics library was picked because they were to some degree developed to simply display things on a web page and are typically much more flexible in terms of what platforms can use it. The choice of picking Three.js on the other hand is mainly because of its simplicity and abundance of resources available to guide during the development of this project.

An additional feature of this part is the two-way communications between the visualizer and the main application which allows for new reconstructions to be immediately viewed when it is ready. The benefits of doing so would be a more interactive and convenient way of asserting the produced reconstructions are of good quality. While it is possible to achieve this with other software such as MeshLab and COLMAP's visualizer, those usually require many file import-export processes which can be time consuming.

Setting up this mode of conversation is not so straightforward however with the typical level of security being put in a web-based application. A solution to this is to set up a web socket connection between the python application and the visualizer. To accommodate the two-way communication, the python application is layered with Flask, a python framework for doing backend servers. As such, the app talks to the visualizer through the web-socket and the server communicates to the app trough exposed endpoints. Note that it is possible to have the communications done purely with the socket. But, setting up the system only with the web socket may leads to a rather complicated form of data processing by the visualizer. This is mainly due to how web sockets, and sockets in general, are typically handled. To keep it simple, this was avoided. Figure 4 shows the communication model between the application and visualizer.

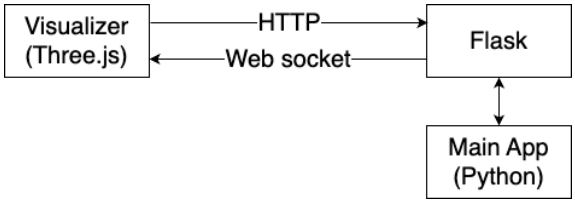


Figure 4: System Communications

2.5 Neighborhood Optimization

One last aspect to be discussed is the optimization problem of a node i.e. How to improve a node given more information like a better set of images. The main idea is to perform reconstruction on the node again using the new set of images.

Simply reconstructing the node however, will not completely solve the problem as parts of the target node also exist in the neighboring nodes. As described in section 2.2.2, to reconstruct a node a reference node is needed and that would apply to the optimization case as well. If the chaining reconstruction approach is to be followed, then during the optimization of a node, it's neighbors would be fixed. The issue with this lies in the fact that some parts of the target node exist in it's neighbors within the overlapping region. If the neighbors are fixed and the node is reconstructed, these overlapping regions will not be optimized since it's also a part of the nodes which are fixed. This leads to the conclusion that if a node is to be optimized, it's neighbors must all also be optimized. The set of nodes that comprises of a target node and it's neighbors is termed as a 'neighborhood' in this project.

Following the discussion, the goal then is to reconstruct a neighborhood and find a reference node for this reconstruction. Reconstructing the neighborhood can be done by combining all set of images within the neighborhood together. To get the reference node, a node that is not within the neighborhood but is connected to the neighborhood is required. In this case, the 'indirect neighbor' of the target node can be of use. An indirect neighbor is a node that is a neighbor to the target node's neighbor but is not a neighbor to the target node. This indirect neighbor can solve the problem raised previously since it will not have any overlaps with the target node but can still act as a reference node. Posed this way, the problem of optimization and chaining reconstruction are actually the same differing only in the fixed nodes and the nodes to be reconstructed. The project's implementation also follows this; chaining reconstruction only has 1 fixed model and 1 reconstructed model while optimization can vary both. Figure 5 illustrates this difference with the blue nodes indicating fixed, red for to be reconstructed, and the gray nodes are not involved.
Do note that, with the current proposed idea, chaining reconstruction should only have 1 fixed model and no more. This is due to the fact that the set of fixed reconstructions need to be set in place using their alignment information which is a relative quantity. During optimization, this is straightforward as everyone involved can use the target node of the optimization to position themselves. During the chaining reconstruction process however, the target node is yet to be reconstructed and in order to position the reconstructions, the node need to position themselves trough some other common node. While it may exist, that node may also be far away. This could be inefficient as the scene captured gets bigger and bigger. Nonetheless, this does not seem to be a major problem as the target

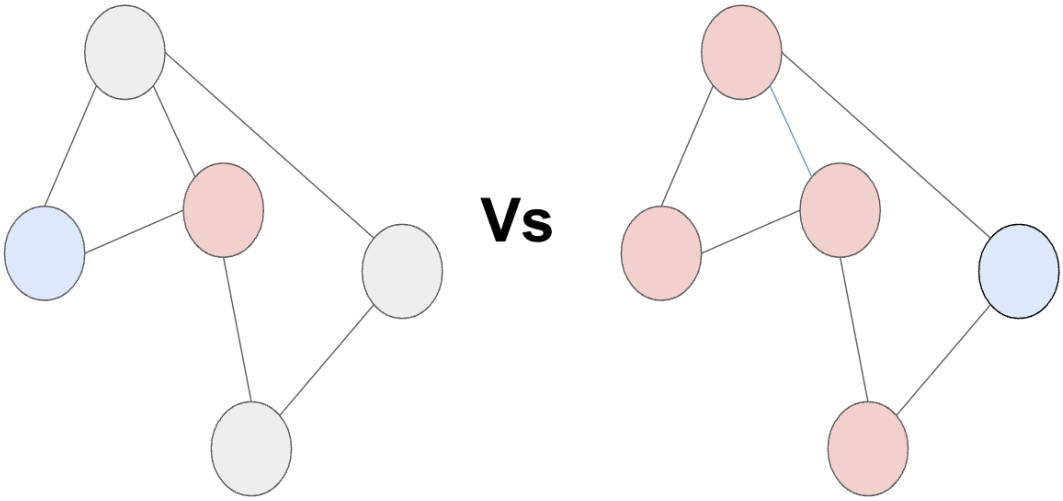


Figure 5: Chaining Reconstruction case (left) and Neighborhood Optimization (right). Blue: fixed, red: to be reconstructed, grey: not involved

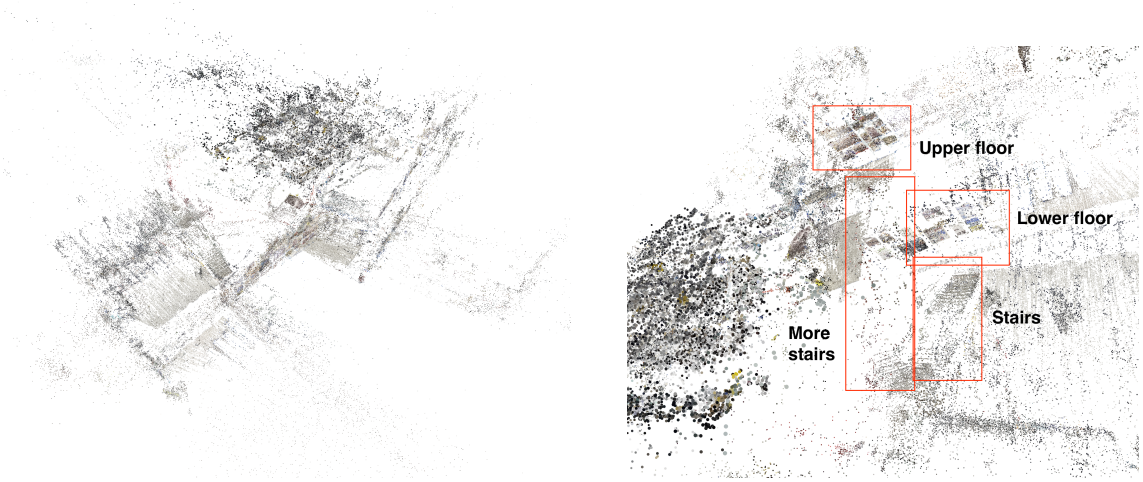
node can be reconstructed with just one of the neighbors. Even though having more fixed nodes can lead to a better reconstruction since there will be more pictures involved, an optimization process can be run after on that newly reconstructed node to get the same effect. An alternative is to not use alignment information and strictly use local translation vector. This however would lead to more complications in the system, less flexibility, and is unjustified since the other approach can achieve similar things.

3 Results

The following sections present results and analyses from chaining reconstructions on bigger scenes done on a non-optimal computer to run COLMAP e.g. Non-CUDA devices. Throughout this section, the approach proposed by this project is at times mentioned as the 'chained reconstruction' approach or simply the 'chained' approach. The usual way of using COLMAP is termed the 'normal' approach.

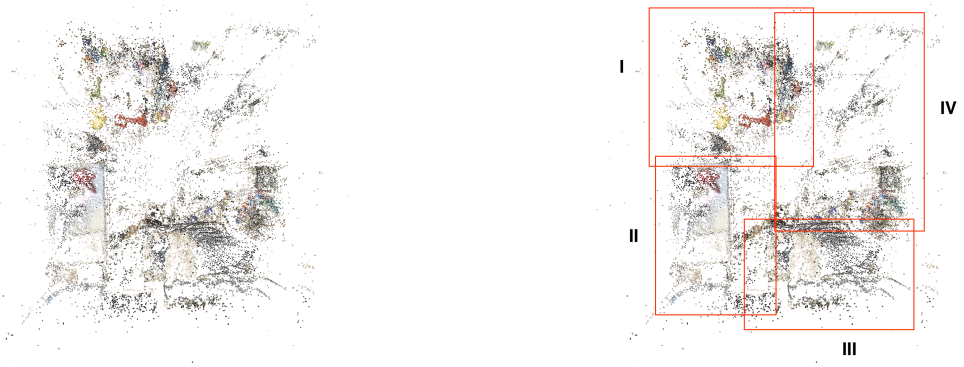
3.1 Reconstructions

The project was run using a Macbook Pro laptop with Apple's M2 chip. All images and videos were captured using only an iPhone 15 camera, without any extra tools. Additionally, no special techniques were involved in the data procurement process. As mentioned in section 1.4, one objective of the project is to capture Innowing. Figure 6a shows its point cloud in its entirety while 6b annotates parts of the Innowing point cloud. The whole process of scanning Innowing took around 9 hours including data gathering. Furthermore, just to show that the other end of the spectrum is achievable, figure 7 shows the point cloud of a corner double room in Jockey Club Student Village III, Lap-Chee College. Figure 7b visualizes how the room is split and figure 7c presents the reconstruction of the normal approach.

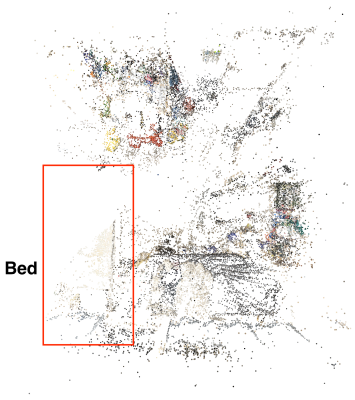


(a) Global view (b) Annotated parts

Figure 6: Innowing point cloud



(a) Global view (b) Annotated parts



(c) Normal Reconstructions

Figure 7: Dorm Room Point cloud

3.2 Discussion

3.2.1 Overview of Results

The reconstruction of the different scenes here shows that the project has managed to provide a solution to the problem posed in section 1.2. A simple implementation was achieved where most of the work was done on data management and the manipulation of functionalities provided by COLMAP. The project was also able to perform reconstruction without needing the whole scene to first be available. The capture of Innowing was done by getting a video of a room, performing the reconstruction chaining for that room, then getting the next room. This is repeated until most of the rooms in Innowing was captured and reconstructed.

3.2.2 Memory Efficiency

An interesting aspect of doing reconstruction this way, is that it has the potential to be faster for bigger scenes while saving on memory usages. For example, the aforementioned bundle adjustment stage for COLMAP may process the entire point cloud of the big scene when all of its components are processed at once which could lead to a higher memory usage. However, all of those points may not be necessary in order to successfully reconstruct the scene. Section II of dorm room (see figure 7b) for example should not need any points within section IV to perform reconstruction since it does not contain any elements from section II. What the proposed approach is doing differently here then is to skip the 'redundant' points. As a result for doing so, the memory usage during the reconstruction can be lowered while still producing a reasonable reconstruction.

To be empirical, figure 8 shows three graphs each plotting memory usages in the reconstructions of different scenes. The memory_profiler python package was used to check for memory usages every 0.1 seconds. Two of the datasets are from KITTI-360 [13] and one of them is the dorm room scene. Particularly, the top graph is the bonsai dataset from KITTI-360, middle one is the dorm room scene, and the last one is the garden scene also from KITTI-360. For the chained COLMAP method, all rooms were split into 4 sections. The y-axis of the graphs specifies the memory usages in MiB and the x-axis specifies the time in seconds unit. The bolder color lines in the graphs represents the chained approach while the more opaque color was attained with the normal approach. The legends on the top corner of each graph also indicate this. The dashed lines signifies the maximum memory usage for the chained reconstruction approach. These graphs reasonably show the efficient memory usage of the proposed approach for each dataset. The plots attained by chaining reconstructions have a lower overall memory usage in comparison for all three scenes supporting the argument made earlier.

Additionally, the graphs illustrate an important aspect during data gathering especially on why the set of images for a particular scene should contain a lot of information while keeping the set small, and that the sizes of each component should ideally be similar. These constraints aim to lower the maximum memory usage while reconstructing each component. A smaller set could lower the maximum memory used as argued before and similar sizes means the minimum images required for each scenes would also be similar. Under these conditions, the chained approach may then be considerably more memory efficient than the normal approach. An extreme example of when the constraints are broken is when the big scene is treated as only one room. In this case the method would return to the normal approach and the overall performance would be equal if not worse.

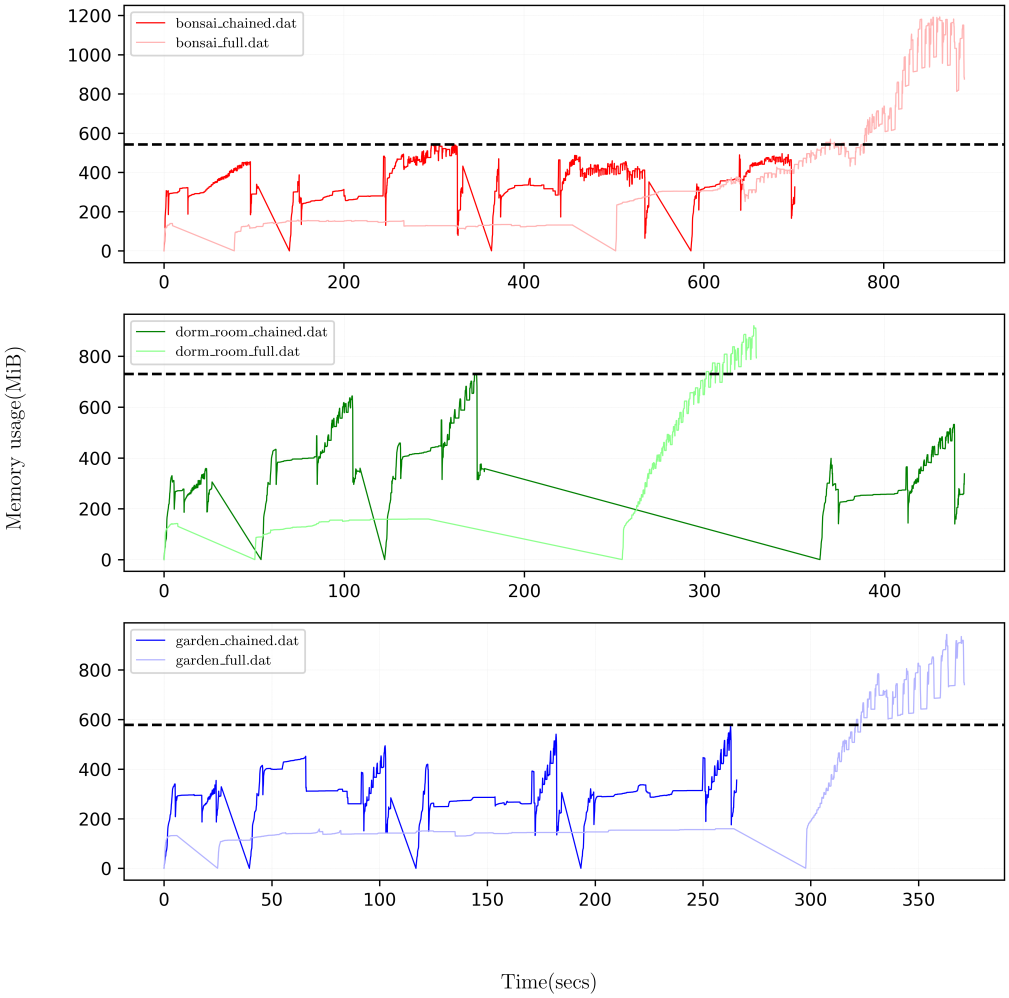


Figure 8: Memory Usages. The darker color indicates the chained approach and the lighter ones represent the usual approach.

3.2.3 Time Efficiency

The argument from before could also explain why chaining may allow for a faster reconstruction process. There is a stage in COLMAP where images are paired to find where objects located in one image is in the other image. Excluding rooms which are not necessary in reconstructing a scene allows for certain pair of images to be skipped and may result in shorter reconstruction time. Take component II of the dorm room scene as an example. It does not need component IV's images for the same reasons as argued previously for memory efficiency and should be skipped. Stopping the comparison can save a considerable amount of time since much less comparisons are done. If the comparisons are not skipped, each image within IV needs to be compared to each image of not just II but also I, and III and those section's images with each other as well. To show that this skipping can save time, figure 8 also shows the total duration of reconstructing each dataset and the chained approach indeed tends to take less time.

Here, one thing to consider is in regards to application bottleneck and how it may affect run-time execution. Figure 8 may seems to show that the chained approach is faster but, this is not true for the dorm room scene. To explain, while profiling the chained approach, the main program was actually re-run every time a new room is reconstructed. This means, every reconstructions of a room during the profiling of the chained approach, comes with the cost of overheads in different parts of the program. For instance, the main program has to connect to a MYSQL database whenever it first runs. For the dorm room scene, there was some significant delay waiting for connection to the DBMS which contributed to the overall time. Eliminating this bottleneck can improve performance and is indeed the case when the app is run normally (a restart is not required after every reconstruction). The reason why the app was re-run every time is due to the fact that COLMAP can randomly crash during it's execution. A safer approach then is to 'save' every time a new reconstruction is made but this does come with the bottleneck. In the case of dorm room, a more typical environment would lead to the same results as the other two datasets as shown in figure 9. Overall, the chained approach indeed seems to be faster in comparison when the cost of the bottleneck is small.

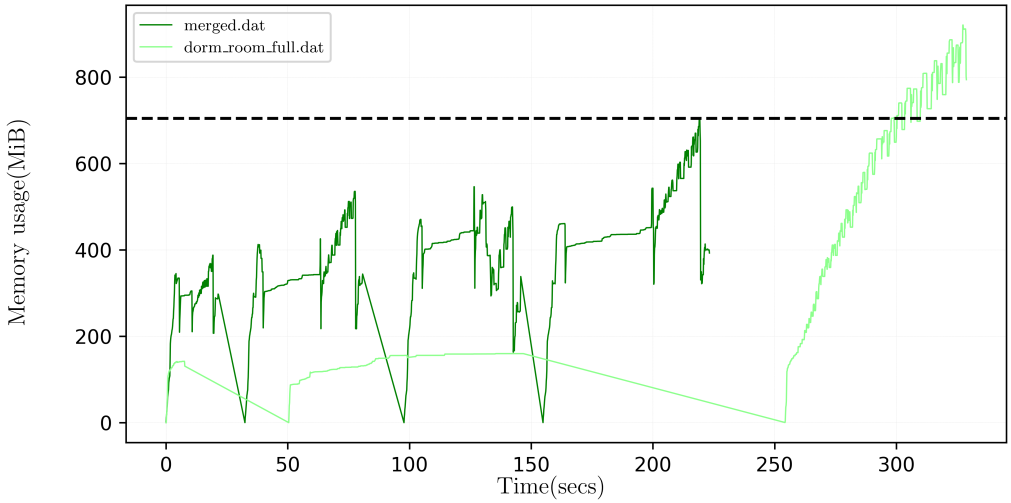


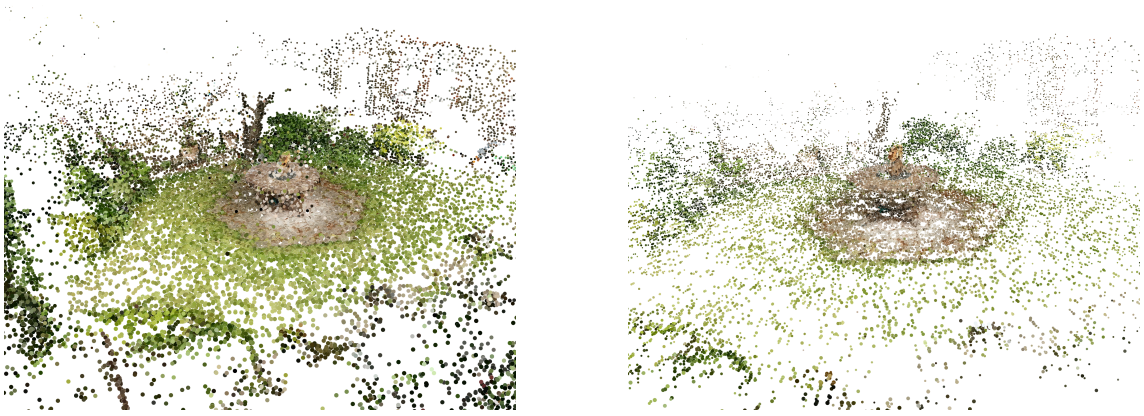
Figure 9: Dorm Room Memory Usage In Normal Conditions

The arguments and results presented so far would suggest the importance of exploiting neighboring information during reconstructions. That is, instead of directly dealing with the scene as a whole, processing it component by component seems to bring a considerable amount of benefits with little costs. It is not just about enabling weaker computers to perform big reconstruction tasks but the method also tends to offer better performance overall.

3.2.4 Importance of A Good Splitting

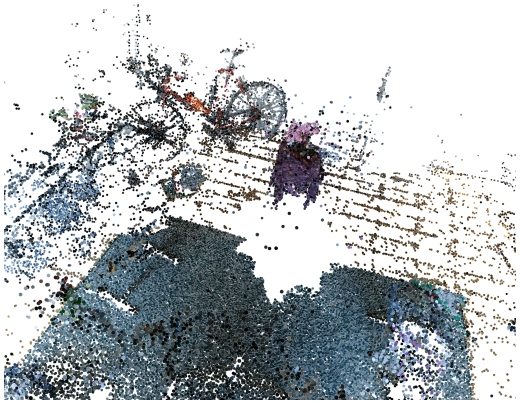
While the results shown so far has been promising, the chaining approach seems to struggle significantly when the components are not properly split. The method puts heavy emphasis on neighboring information and so if this is not satisfied, a low quality reconstruction may be produced. Figure 10, 11, and 7 shows the reconstruction of garden, bonsai, and the

dorm room scene respectively along with how they would look like if the normal COLMAP pipeline is applied. The garden scene looks somewhat similar to the output produced by the normal approach but more sparse while the bonsai scene is barely recognizable. The cases of the garden and bonsai scenes can be explained due to the bad splitting done for these two. The provided dataset for both only contain image folders for the scenes without any explicit splitting. The breaking down of a scene in this case is done by randomly taking the images and sorting them by their name and then taking roughly equal portions of that collection making each portion a 'scene'. This haphazard splitting would mean the chained reconstruction approach may struggle to exploit neighboring information as some images that should overlap are not included together during the reconstruction process. This way, the output may come out considerably worse than the expected outcome since there could be less correspondence found between images which leads to a lesser amount of triangulated points. The dorm room scene on the other hand is different as its splitting was planned. As a result, a better reconstruction could be achieved. This is demonstrated by comparing figure 7c and figure 7b especially around the bed parts as annotated. The overall reconstruction produced by the chaining the reconstructions were also denser by having around 9000 more points within the overall point cloud (The normal approach has 27850 points while the chained approach produced 37500 points). Perhaps, if the dataset of garden and bonsai was better split, a better results could be achieved as in the case of the dorm room dataset.



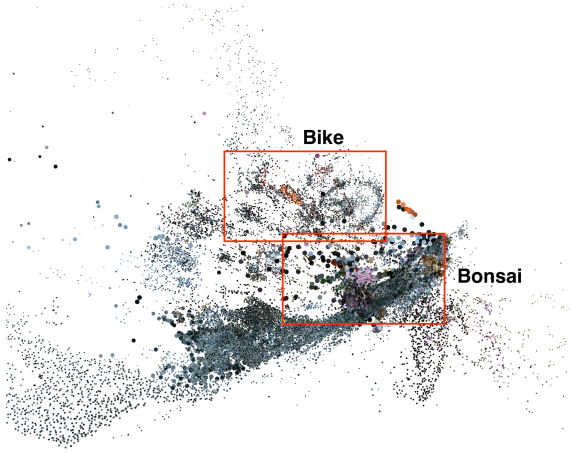
(a) Normal Reconstruction (b) Chained Reconstruction

Figure 10: Garden Reconstructions



(a) Dense Reconstruction

(b) Normal Reconstruction



(c) Chained Reconstruction

Figure 11: Bonsai Reconstructions

3.2.5 Comparison With Other Approaches

Comparing the approach proposed in this paper is a little difficult since either it's not fair or the body of work has no published code to benchmark against.

It is not fair to compare the approach proposed by this project and that of BlockNeRF [10] and Hierarchical Gaussian Splatting [9] since the current approach excludes scene rendering. While the logical comparison would be with the works of Zhu et al. [7] and Qu et al. [8], they unfortunately did not provide any source code.

However, the one aspect that can be compared is on the rendering of the components as to how they would look like when combined together as shown in figure 12. The outcome is rather poor but this is to be expected. BlockNeRF and Hierarchical Gaussian Splatting spend some time in dealing with the regions of which the scene is cut and how to consolidate them together. With that being the case, if the components of the output of the project is to be rendered and combined together, some care has to be given on how to merge the rendering of the overlapping region. However, this does not mean the output of the project is unusable. The components can still be combined before being rendered. Though perhaps the computer that is doing the rendering may need to be a different,

more optimal machine for the task.

Figure 12: Merged Gaussian Splats of Innowing

4 Limitations

4.1 Machine Limitation

Although the approach taken here allows for a better performance in processing big scenes, there are still some difficulties faced in aspects regarding COLMAP and data gathering. For example, while COLMAP can utilize GPU cards, it's capabilities without CUDA is limited as mentioned. One of the limitations is the restrictions of the number of matches. This brings great difficulties in a scene with many potential features. Those types of scenes are limited to what they can contribute to the reconstruction. For example, the flight of stairs in figure 6b is missing a lot of details on the stairs themselves because of this restriction. They have too many matches on other parts of the scenery instead. At times, it would not even be possible to continue the reconstruction process since the portion of the overlap might also get skipped. To workaround this, those complex scenes can be reconstructed by a stronger computer. Another alternative is to simply brute force it and cut those scenes into even smaller parts with each part focusing on a simpler portion of the scenery. The goal here is to make each part contribute to the main reconstruction no matter how small. Eventually, if each part can make some contributions, the whole scenery can be reconstructed. This was the approach taken in this project.

4.2 Depth of A Scene

A natural way to split a scene is by re imagining it into rooms. However, there are some scenarios which can make this decision to be complicated. An example is when there is a hallway near the room of interest. This becomes quite tricky as the captured images may also reconstruct parts of the hallway which are far away from the room of concern. The management of the neighboring information then becomes a bit more chaotic to handle since bounding a room now involves double checking the taken video/image. To deal with this, given a scene, after splitting it into rooms, never let the hallways or a faraway scene

to enter the angle of capture. This however, may not be possible and a compromise has to be made where everything involved is captured. For instance, hallways, especially long, empty, narrow hallways, may force this to happen. Innowing has these types of rooms and as a result, some components of it are bigger than others.

4.3 Margin of Error In Previous Reconstructions

As the reconstruction gets bigger and bigger, reconstructing a new portion can get a bit more difficult. This may be due to the accumulating errors of the previous parts. If traditional COLMAP is used, the 'previous parts' can freely move and adjust giving more flexibility in the face of bad correspondence between images (which can happen with SIFT. SIFT is a feature detector used by COLMAP). A solution is to swap the feature detector with something more robust. However as long as the feature detector itself is not perfect, this issue with the proposed approach of splitting reconstructions would be difficult to get rid of. One way that worked for this project is to make sure to take as many images in different perspectives of the overlapping region between models to make sure that region is as detailed as possible. This overlapping region is one of the determining factor on whether or not the new model captured will show up in the main reconstruction. Another way is to make sure the sizes of each scenes are not too small. Bigger scenes can tolerate more error but would also take a longer time to reconstruct and may not even be processable by the present computer.

5 Challenges

In starting the project, a great deal of time was spent in coming up with the idea. First couple of iterations were spent on not chaining at all but to simply move the scenes around according to some camera positions. This is quite difficult due to the uncertainty and scale issues mentioned in section 2.2.2. There was also an issue with how to deal with bad initial reconstruction which lead to the neighborhood optimization introduced in section 2.5.

Understanding the COLMAP code base particularly with the way a 3d reconstruction is stored was also a challenge. COLMAP does not store reconstructions within their database. Rather it is stored inside three separate files each describing a different quantity. Learning how these files work and their overall dynamics to the COLMAP pipeline was not straightforward. A considerable amount of time was spent here.

Another challenge comes from designing the application. There were many issues regarding how the system should be modeled out and on its efficiency. The inexperience in building a computer vision application was a significant factor to these difficulties. At times, incorrect decisions were made on what class should be implemented, how the flow of data throughout the application should be, etc. These were problems of which the answers to were found by extensive trials and errors. Still, In the end, much knowledge was gained.

6 Conclusion

Recent advancements within the field of inverse graphics has allowed for numerous research directions particularly in the domains of processing large scenes. Previous works had described methods that while can produce great results, are also convoluted at times

and typically either require entire scenes to be present, or eventually struggle as the scene becomes too big. This report on the other hand describes a straightforward and reasonably efficient approach to deal with the breaking down of a scene without the availability of the entire scene by performing the splitting in the initial reconstruction stages. Having such a system would not only improve efficiency of performing 3d reconstruction processes, but also simplify the process of capturing bigger scenes. Now, not only is the processing done in parts but the data procurement process can also be treated in the same way extending the divide-and-conquer nature of the work within the topic.

These great benefits were argued to be achieved because of the approach's ability to work mostly on neighboring information. That is, since one portion of a room does not have any portion of the other room, that portion should not be involved in the other portion's reconstruction processes. The capabilities of such an idea was shown through numerous reconstructions of various sizes reconstructed using a non-optimal computing device (non-CUDA devices) along with a trace of their memory use. From such reconstructions however, some bad cases of the approach such as dealing with bad splitting and poor merged renders of each reconstructions were identified.

The approach is also not free of limitations. These issues include, but are not limited to, a cap on finding number of matches between images, difficulties in bounding a scene, and the challenges in chaining as the scene gets bigger. Future works on this topic could consider any of the limitations mentioned here. Though, finding a better way to bound a scene would be the recommendation.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[3] Y.-H. Huang, Y.-P. Cao, Y.-K. Lai, Y. Shan, and L. Gao, "Nerf-texture: Synthesizing neural radiance field textures," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–15, 2024.

[4] J. Cen, J. Fang, C. Yang, *et al.*, "Segment any 3d gaussians," *arXiv preprint arXiv:2312.00860*, 2023.

[5] Y. Xie, C.-H. Yao, V. Voleti, H. Jiang, and V. Jampani, "SV4D: Dynamic 3d content generation with multi-frame and multi-view consistency," *arXiv preprint arXiv:2407.17470*, 2024.

[6] H. Xia, Z.-H. Lin, W.-C. Ma, and S. Wang, *Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video*, 2024. arXiv: 2404.09833 [cs.CV].

[7] S. Zhu, T. Shen, L. Zhou, R. Zhang, T. Fang, and L. Quan, "Accurate, scalable and parallel structure from motion," Ph.D. dissertation, Hong Kong University of Science and Technology, 2017.

[8] Y. Qu, J. Huang, and X. Zhang, "Rapid 3d reconstruction for image sequence acquired from uav camera," *Sensors*, vol. 18, no. 1, 2018, ISSN: 1424-8220. DOI: 10.3390/s18010225. [Online]. Available: https://www.mdpi.com/1424-8220/18/1/225.

[9] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A hierarchical 3d gaussian representation for real-time rendering of very large datasets," *ACM Transactions on Graphics*, vol. 43, no. 4, Jul. 2024. [Online]. Available: https://repo-sam.inria.fr/fungraph/hierarchical-3d-gaussians/.

[10] M. Tancik, V. Casser, X. Yan, *et al.*, *Block-nerf: Scalable large scene neural view synthesis*, 2022. arXiv: 2202.05263 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2202.05263.

[11] H. Luo, J. Zhang, X. Liu, L. Zhang, and J. Liu, "Large-scale 3d reconstruction from multi-view imagery: A comprehensive review," *Remote Sensing*, vol. 16, no. 5, 2024, ISSN: 2072-4292. DOI: 10.3390/rs16050773. [Online]. Available: https://www.mdpi.com/2072-4292/16/5/773.

[12] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," *arXiv preprint arXiv:2109.13410*, 2021.